Relatório do Projeto de SRCR

Bruno Veloso (a78352) Jaime Leite (a80757)
João Pimentel (a80874) Rodolfo Silva (a81716)
Pedro Gonçalves (a82313)

Braga, abril de 2019

Universidade do Minho
Mestrado Integrado em Engenharia Informática
Sistemas de Representação de Conhecimento e Raciocínio
(2º Semestre/2018-2019)
Grupo 3

Resumo

O projeto em questão trata-se na explicação do processo de desenvolvimento do segundo projeto prática da unidade curricular Sistemas de Representação de COnhecimento e Raciocínio. O tema do mesmo é a programação em lógica extendida e conhecimento imperfeito.

Deste modo, é requerido o desenvolvimento de um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo relativo a uma área de prestação de serviços de saúde, possuindo utentes, prestadores dos serviços e cuidados de saúde, de diferentes instituições e cidades.

Com o intuito de retratar conhecimento imperfeito, é necessária a recorrência a valores nulos, bem como exceções, de modo a retratar com maior critério cada caso.

Dito isto, o desenvolvimento de diversos predicados auxiliares possibilitou a fácil compreensão e desenvolvimento do projeto proposto, apesar de alguns problemas encontrados ao longo do caminho.

Conteúdo

Lista de Figuras

1 Introdução

Este relatório é resultado da elaboração do segundo projeto prático da unidade curricular Sistemas de Representação de Conhecimento e Raciocínio, abordando o tema da programação em lógica extendida e o conhecimento imperfeito.

Este exercício retrata, tal como no exercício anterior, uma base de conhecimento sobre uma área de prestação de serviços, no entanto, desta vez é necessário ter em consideração a representação de conhecimento imperfeito, recorrendo à utilização de valores nulos.

Existme três grandes fontes de conhecimento, sendo estas os utentes, os prestadores e os cuidados, que relacionam os dois anteriores. O objetivo é, não só, desenvolver um sistema de inferência capaz de responder a questões, mas também permitir que o sistema seja capaz de guardar conhecimento imperfeito passado ao programa de modo a que sejam criados os predicados necessários para indicar possíveis valores de exceção.

Desta forma, nas próximas secções serão abordados os diversos tipos de conhecimento tratados no projeto, explicando em que consistem e as formas como foram retratados no projeto em si.

2 Preliminares

2.1 Base de Dados ou Representação de Conhecimento

Por forma a ser possível lidar com detalhes do mundo que nos rodeia, existem sistemas de bases de dados e sistemas de representação de raciocínio, que têm como principal função manipular tal informação. Os sistemas referidos fazem a distinção entre a descrição dos dados e respetivo esquema conceptual (**representação da informação**) e a obtenção de respostas para certas questões, assente no processo de tratamento dos dados (**computação da informação**).

Sendo assim, são apresentados no texto seguinte os pressupostos que servem de base às linguagens de programação referentes aos sistemas de bases de dados e aos sistemas de representação de conhecimento.

2.1.1 Base de Dados

Neste tipo de sistemas, os pressupostos primordiais referem que somente a informação que está referida na base de dados é verdadeira, tendo tudo o resto valor de verdade contrário.

Pressupostos de base aos sistemas de base de dados:

- **Pressuposto do Mundo Fechado** – a informação não referida na base de dados é considerada falsa;
- **Pressuposto dos Nomes Únicos** – duas entidades diferentes do universo de discurso são designadas por duas constantes diferentes, quando estas últimas definem valores atómicos ou objetos;
- **Pressuposto do Domínio Fechado** – os objetos que são designados por constantes na base de dados são os únicos que fazem parte do universo de discurso.

2.1.2 Sistema de Representação de Conhecimento

Contrariamente ao sistema de base de dados, os sistemas de representação de conhecimento podem, por vezes, não assumir que a informação e entidades que são representadas são as únicas que são consideradas válidas.

- **Pressuposto do Mundo Aberto** – os factos ou conclusões verdadeiros representados na base de conhecimento podem não ser únicos, podendo haver representações para além desta;

- **Pressuposto dos Nomes Únicos** – duas entidades diferentes do universo de discurso são designadas necessariamente por duas constantes diferentes que definam valores atómicos ou objetos;

- **Pressuposto do Domínio Aberto** – os objetos designados por constantes na base de conhecimento podem não ser os únicos do universo de discurso.

Pressupostos de base aos sistemas de representação de conhecimento:

2.2 Representação de Informação Incompleta

Diariamente deparamo-nos com questões onde a informação é impercetível ou até mesmo incompleta. Perante isto, é útil conseguir representar esta mesma informação, para lá do que é preciso, ou seja, daquilo que se pode dizer de imediato se é verdadeiro ou falso. É necessário demonstrar que tais situações são incompletas ou desconhecidas, em oposição ao que acontece, por exemplo, nas linguagens de programação com que lidamos diariamente, em que os objetos têm um valor exatamente conhecido.

Com isto, aparece um tipo de programação, designado por programação em lógica estendida (PLE), capaz de resolver este tipo de situações de informação incompleta.

Uma questão pode assim ter três tipos distintos de resposta:

- **Verdadeiro** – quando for possível, para uma questão(Q), provar a sua veracidade na base de conhecimento;

- **Falso** – quando for possível, para uma questão, provar a sua falsidade(-Q) na base de conhecimento;

- **Desconhecido** – quando não for possível provar, na base de conhecimento, a questão(Q) nem a questão(-Q).

O grupo teve assim de definir outro tipo de informação (para além da positiva e negativa) para cumprir os pressupostos enunciados anteriormente.

Passam a existir assim dois tipos de negação distintos:

- **Negação por falha** – representada pelo teorema *nao* que foi utilizado no trabalho prático transato. Este tipo de negação acontece quando não existe nenhuma prova;

- **Negação forte** - representada pelo teorema (-). Acontece quando se afirma que um dado predicado é falso.

2.3 Valores nulos

No panorama da representação de informação incompleta, valores nulos servem para representar todas as questões desconhecidas, isto é, todas as questões que não são nem verdadeiras nem falsas.

Estes valores nulos possuem três sub-divisões, todas distintas umas das outras. A primeira representa valores desconhecidos; a segunda representa valores desconhecidos, mas que pertencem a um determinado conjunto de valores; por fim, a terceira representa valores não permitidos, isto é, valores interditos na base de conhecimento.

De seguida, vão ser explicados de forma mais extensiva estas syb-divisões dos valores nulos.

2.3.1 Desconhecido

Tal como o nome indica, este tipo de valor nulo representa algo que não é conhecido, isto é, não existe nenhuma prova de que algo é verdadeiro ou falso. Se tomarmos como exemplo o predicado pai(X,João), se o João não souber quem é o seu pai, a resposta ao mesmo, nestas circunstâncias tem de ser desconhecida, pois apesar de o João ter um pai é impossivel saber em concreto quem esse pai é.

2.3.2 Impreciso

Este segundo tipo de valor nulo é muito semelhante ao tipo discutido anteriormente, apenas com a diferença que o conhecimento só é desconhecido numa determinada gama de valores. Ou seja, se um valor não pertencer a essa dada gama, esse predicado será falso, caso contrário será desconhecido pois apenas se sabe que o valor pertence a essa gama, mas não se sabe qual o valor ao certo.

2.3.3 Interdito

O terceiro e último tipo de valor nulo é um pouco diferente dos apresentados anteriormente, uma vez que, para além de identificarem valores desconhecidos, definem também um tipo de dados que não é admissível existir na base de conhecimento.

Um exemplo bastante prático é retratado na famosa série televisiva "A Guerra dos Tronos", onde Jon Snow não pode saber quem são os seus pais. Quer isto dizer que os valores que representam estes são desconhecidos e

assim devem permanecer, não sendo possível atribuir-lhes um valor real e saber que esse valor é verdadeiro.

3 Descrição do Trabalho e Análise de Resultados

Como mencionado nos capítulos anteriores, foi proposta a realização de um projeto que abordasse a temática do conhecimento imperfeito. Desta forma, neste capítulo serão retratados todos os requisitos apresentados pelo docente, bem como as formas de conceção das soluções dos mesmos, de uma forma clara.

3.1 Fontes de Conhecimento

Pretende-se representar um universo de prestação de serviços médicos, como tal existem três grandes fontes de conhecimento: utentes, prestadores e cuidados. Esta última relaciona os utentes com os prestadores, na medida em que representa o ato do serviço prestado.

Além destes três predicados, existem predicados auxiliares que servem para ajudar a retratar conhecimento imperfeito, como, por exemplo, *incerto_ idade*, que associa o identificador de um utente ao valor que representa a incerteza na idade do utente em questão, *impreciso* que indica qual o predicado com uma imprecisão, através do seu identificador, e *nulo* que indica que o valor é relativo a conhecimento interdito. No entanto, estes predicados serão apresentados detalhadamente nos próximos capítulos.

Dito isto, seguem-se as definições das fontes de conhecimento retratadas:

- utente: (#IdUt, Nome, Idade, Cidade)
- prestador: (#IdPrest, Nome, Especialidade, Instituicao ,Cidade)
- cuidado: (#Id, Data, #IdUt, #IdPrest, Descricao, Preco)

3.2 Representação de Conhecimento Positivo e Negativo

No trabalho prático anterior, já foram retratados exemplos de conhecimento positivo, tendo sido utilizados como base deste projeto, após algumas alterações necessárias devido ao novo panorama. Na Figura 1 tem-se o exemplo disto no que toca aos utentes.

```
utente(1,joao_pimentel,20,esposende).
perfeito(utente(1)).

utente(2,bruno_veloso,22,ponte_da_barca).
perfeito(utente(2)).

utente(3,jaime_leite,20,felgueiras).
perfeito(utente(3)).

utente(4,pedro_goncalves,20,felgueiras).
perfeito(utente(4)).

utente(5,rodolfo_silva,20,trofa).
perfeito(utente(5)).

utente(6,joana_amorim,22,viana_do_castelo).
perfeito(utente(6)).

utente(7,gloria_borga,53,portimao).
perfeito(utente(7)).

utente(8,andreia_silva,27,trofa).
perfeito(utente(8)).
```

Figura 1 - Conhecimento Positivo.

Contudo, também foi necessária a adição de conhecimento negativo, sendo que a importância deste é enorme, pois retrata um conhecimento perfeito só que com valor negativo. Este tipo de conhecimento foi representado por dois tipos: negação por falha e negação forte, como se vê na Figura 2.

```
% Negação Por Falha
-prestador(Id,Nome,Especialidade,Instituicao,Cidade) :-
  nao(prestador(Id,Nome,Especialidade,Instituicao,Cidade)),
  nao(excecao(prestador(Id,Nome,Especialidade,Instituicao,Cidade))).

% Negação Forte
-prestador(11, veronica_silva, imunoalergologia, publico, braganca).
perfeito(prestador(11)).
```

Figura 2 - Negação Forte e Negação Por Falha.

3.3 Representação de Conhecimento Imperfeito

A utilização de valores nulos surge como uma forma de distinguir diferentes situações, isto é, distinguir entre situações em que as respostas às questões são apresentadas como conhecidas, podendo ser verdadeiras ou falsas, ou quando são, simplesmente, desconhecidas.

Tendo em conta a existência de três tipo de conhecimento imperfeito (incerto, impreciso e interdito), será explicado, separadamente, o processo para representar cada um destes tipos de conhecimento.

3.3.1 Conhecimento Incerto

Este tipo de conhecimento é utilizado quando se fala de um determinado valor nulo e não necessariamente de um conjunto de valores. A título de exemplo, se não se conhecer a morada da utente Maria de Lurdes, será indicado, através do predicado *incerto_morada*, que o valor associado a esta utente (*nulo2*) é um valor desconhecido, tendo que ser criada uma exceção para o mesmo (Figura 3).

```
cuidado(8,04-04-2017, 1, 2, consulta_rotina, nulo1).
excecao(cuidado(Data, IdUt, IdPrest, Descricao,Custo)) :-
    cuidado(Data, IdUt, IdPrest, Descricao, nulo1).
incerto_preco(cuidado(8,nulo1)).

utente(18, maria_de_lurdes, 68, nulo2).
excecao(utente(Id, Nome, Idade, Morada)) :-
    utente(Id, Nome, Idade, nulo2).
incerto_morada(utente(18,nulo2)).

utente(16, jose_manuel, nulo5, braga).
excecao(utente(Id, Nome, Idade, Morada)) :-
    utente(Id, Nome, nulo5, Morada).
incerto_idade(utente(16,nulo5)).

prestador(7, mario_quintas, nulo7, clipovoa, povoa_varzim).
excecao(prestador(ID, Nome, Especialidade, Instituicao, Cidade)) :- prestador(ID, Nome, nulo7, Instituicao, Cidade).
incerto_especialidade(prestador(7,nulo7)).
```

Figura 3 - Conhecimento Incerto.

3.3.2 Conhecimento Impreciso

Neste caso, o valor nulo pode estar contido dentro de uma gama de valores previamente determinados, não sendo conhecido qual o valor exato que concretiza a questão. Quer isto dizer que, como se vê na Figura 4, ao registar o utente 14, o senhor Alfredo, não se percebeu se ele morava em Braga ou na Barca, sendo criada uma exceção para cada um destes casos e uma indicação de que este utente possui conhecimento impreciso.

```
excecao(utente(14, alfredo, 74, braga)).
excecao(utente(14, alfredo, 74, barca)).
impreciso(utente(14)).

excecao(cuidado(9,31-12-2019, 7, 6, eletrocardiograma, X)) :- X > 20, X < 30.
impreciso(cuidado(9)).
```

Figura 4 - Conhecimento Impreciso.

3.3.3 Conhecimento Interdito

Já neste tipo de conhecimento, o valor nulo, além de identificar um valor desconhecido, não permite especificar ou conhecer o seu valor real. Quer isto dizer que qualquer tentativa para concretizar este deverá ser rejeitada. Por exemplo, por motivos de segurança, não se pode conhecer a morada do utente Roberto Leque. Assim, é criada uma exceção relativa ao valor nulo, de modo a mostrar que este é desconhecido, este é indicado como sendo interdito (predicado *nulo*) e é definido um invariante que não permite a re-definição do valor em questão.

```
utente(15, renato_sancho, nulo4, moinhos).
excecao(utente(Id, Nome, Idade, Morada)) :-
    utente(Id, Nome, nulo4, Morada).
nulo(nulo4).
+utente(A,_,_,_) :: (
  solucoes(A,(utente(A,_,nulo4,_),nulo(nulo4)),B),comprimento(B,0)).

utente(17, roberto_leque, 32, nulo6).
excecao(utente(Id, Nome, Idade, Morada)) :-
    utente(Id, Nome, Idade, nulo6).
nulo(nulo6).
+utente(A,_,_,_) :: (
  solucoes(A,(utente(A,_,_,nulo6),nulo(nulo6)),B),comprimento(B,0)).
```

Figura 5 - Conhecimento Interdito.

3.4 Invariantes que Restringem a Inserção e Remoção de Conhecimento

De modo a que a base de conhecimento funcione de forma apropriada e correta, é necessário implementar uma série de invariantes que serão responsáveis por controlar a inserção e remoção de conhecimento.

3.4.1 Inserção

3.4.1.1 Utentes

A primeira preocupação é garantir que não se pode inserir conhecimento perfeito sobre um utente que já possui este tipo de conhecimento. Como a indicação de conhecimento perfeito é relativa ao identificador de um utente, este invariante garante, ainda, a não repetição deste, ou seja, a não existência de conhecimento repetido (Figura 6).

Em seguida apresenta-se um invariante que não permite inserir conhecimento impreciso se já existir perfeito ou impreciso. Note-se que, como o impreciso é mais concreto que o incerto, é possível substituir o conhecimento incerto por impreciso. O conhecimento impreciso apenas pode ser substituído por conhecimento perfeito positivo, já que este apenas possui um valor desconhecido, dentro de uma gama de valores.

No caso de inserção de conhecimento incerto ou interdito, esta não pode ser efetuada se já existir conhecimento perfeito, impreciso ou incerto, daí a necessidade do terceito invariante na Figura 6.

Por fim, no caso de inserção de conhecimento perfeito negativo, apenas se confirma que não existe nenhuma referência perfeita, positiva ou negativa, ao utente em questão.

```
% Utente

% Invariantes que nao permitem inserir conhecimento perfeito
% se ja existir conhecimento perfeito

+utente(IdUt, Nome, Idade, Morada) :: (
  nao(perfeito(utente(IdUt)))
).

% Invariante que nao permite inserir conhecimento impreciso se
% ja existir conhecimento perfeito ou impreciso

+utente(IdUt, Nome, Idade, Morada) :~: (
  nao(perfeito(utente(IdUt))),
  nao(impreciso(utente(IdUt)))
).

% Invariante que nao permite a insercao de conhecimento
% incerto/interdito se ja existir conhecimento.
+utente(IdUt, Nome, Idade, Morada) :-: (
  nao(perfeito(utente(IdUt))),
  nao(impreciso(utente(IdUt))),
  nao(incerto(utente(IdUt)))
).

% Invariante que nao permite a insercao de conhecimento
% negativo se ja existir conhecimento perfeito
+(-utente(IdUt, Nome, Idade, Morada)) :: (
  nao(perfeito(utente(IdUt)))
).
```

Figura 6 - Invariantes sobre Inserção de Utentes.

3.4.1.2 Prestadores

Tal como no caso dos utentes, os dois primeiros invariantes na Figura 7 mostram que, caso já exista conhecimento perfeito na base de conhecimento, não se pode sobrepor este conhecimento. Os dois invariantes restantes seguem, também, a mesma lógica dos utentes, onde conhecimento impreciso não pode sobrepor perfeito nem impreciso já existente, nem incerto/interdito devem sobrepor conhecimento, caso este já exista.

```
% Prestador

% Invariante estrutural: nao permitir a insercao de conhecimento repetido
+(-prestador(IdPro, Nome, Especialidade, Instituicao, Cidade)) :: (
  nao(perfeito(prestador(IdPro)))
).


% Invariantes que nao permitem inserir conhecimento perfeito
% se ja existir conhecimento perfeito

+prestador(IdUt, Nome, Especialidade, Instituicao, Cidade) :: (
  nao(perfeito(prestador(IdUt)))
).

% Invariante que nao permite inserir conhecimento impreciso se
% ja existir conhecimento perfeito ou impreciso

+prestador(IdUt, Nome, Especialidade, Instituicao, Cidade) :~: (
  nao(perfeito(prestador(IdUt))),
  nao(impreciso(prestador(IdUt)))
).

% Invariante que nao permite a insercao de conhecimento
% incerto/interdito se ja existir conhecimento.
+prestador(IdUt, Nome, Especialidade, Instituicao, Cidade) :-: (
  nao(perfeito(prestador(IdUt))),
  nao(impreciso(prestador(IdUt))),
  nao(incerto(prestador(IdUt)))
).
```

Figura 7 - Invariantes sobre Inserção de Prestadores.

3.4.1.3 Cuidados

No que toca aos cuidados, um cuidado apenas pode ser adicionado em caso de quero utente, quer o prestador existirem na base de conhecimento (Figura 8). No entanto, como podem existir cuidados com utente/prestador incerto/interdito, foram definidos dois invariantes específicos para estes casos, sendo estes o segundo e terceiro da Figura 8.

Os restantes invariantes seguem a mesma lógica dos utentes e prestadores, explicados anteriormente.

Figura 8 - Invariantes sobre Inserção de Cuidados.

3.4.2 Remoção

3.4.2.1 Utentes

No caso de remoção dos dados de um utente, é necessário confirmar que este não possui cuidados a si associados. Caso fosse removido e possuísse cuidados, a base de conhecimento deixaria de ser consistente.

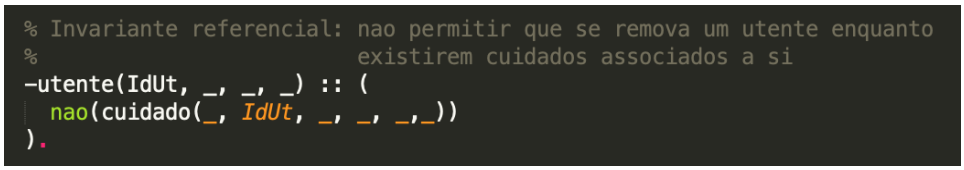

Figura 9 - Invariantes sobre Remoção de Utentes.

3.4.2.2 Prestadores

Tal como no caso dos utentes, caso se remova um prestador que possua cuidados associados a si mesmo, a base de conhecimento passaria a estar

incoerente, pelo que deve ser confirmada a inexistência dos mesmos antes da remoção do registo.

```
% Invariante referencial: nao permitir que se remova um utente enquanto
%                         existirem atos medicos associados a si
-prestador(IdUt, _, _, _,_) :: (
  nao(cuidado(_, _, IdUt, _, _,_))
).
```

Figura 10 - Invariantes sobre Remoção de Cuidados.

3.5 Problemática da Evolução de Conhecimento

O tratamento da problemática da evolução de conhecimento prende-se com o facto de manter a base de conhecimento coesa e inviolável em termos de existência de conhecimento repetido, tendo em conta cada inserção ou remoção que possa acontecer.

De modo a garantir esta segurança, não pode ser apagada informação dependente de outra, ou seja, não pode ser removida a informação sobre um utente se este possuir consultas a si associadas. Além disso, não pode ser adicionada informação repetida, pois não traz adição de conhecimento. Assim, no momento de alteração de informação, é necessário testar se esta corrompe a base de conhecimento. Estes testes são efetuados através do uso de invariantes, previamente explicados.

Tendo em conta a existência de vários tipos de conhecimento, em que varia o que será adicionado à base de conhecimento, foram definidos vários predicados relativos a cada caso em específico. Por exemplo, na Figura 11, vê-se um exemplo de inserção de conhecimento perfeito positivo sobre um utente, em que se remove qualquer conhecimento imperfeito que já exista, pois este é verdadeiro, sendo tudo o resto falso, para este caso.

```
evolucao_perfeito(utente(IdUt,Nome,Idade,Morada)) :-
  solucoes(Inv, +utente(IdUt,Nome,Idade,Morada)::Inv, LInv),
  testa(LInv),
  remover_impreciso(utente(IdUt,Nome,Idade,Morada)),
  assert(utente(IdUt,Nome,Idade,Morada)),
  assert(perfeito(utente(IdUt))).
```

Figura 11 - Evolução Perfeita Positiva de Utente.

No caso de o conhecimento ser impreciso, ou seja, um certo valor não ser completamente desconhecido, têm que ser criadas exceções com os respetivos valores em estudo. Assim, como se vê na Figura 12, é confirmado que se está a tratar do mesmo prestador (confirmando o seu identificador), são testados os invariantes relativos ao conhecimento em questão, é removido qualquer conhecimento incerto que possa existir na base de conhecimento relativo a este caso em específico e, por fim, são adicionadas as exceções que permitirão o sistema de inferência indicar que, dentro dos valores inseridos, se trata de um caso desconhecido.

```
evolucao_impreciso([prestador(IdUt, Nome, Especialidade, Instituicao, Cidade)|T]) :-
  T \= [],
  same_prestador(T, IdUt),
  testaInvs([prestador(IdUt, Nome, Especialidade, Instituicao, Cidade)|T]),
  remover_incerto(prestador(IdUt, Nome, Especialidade, Instituicao, Cidade)),
  insere_excecoes([prestador(IdUt, Nome, Especialidade, Instituicao, Cidade)|T]).
```

Figura 12 - Evolução Imprecisa de Prestador.

No que toca à inserção de conhecimento incerto, o processo é semelhante, sendo que, primeiramente, é garantido que não existe conhecimento sobre este utente. Caso passe nos testes, é inserida uma exceção com o valor nulo associado à morada do utente, o utente é inserido na base de conhecimento e uma indicação da incerteza no valor da sua morada fica registada, como se vê na Figura 13.

```
evolucao_incerto_morada(utente(IdUt, Nome, Idade, Morada)) :-
  solucoes(Inv, +utente(IdUt,Nome,Idade,Morada):-:Inv, LInv1),
  solucoes(Inv, +utente(IdUt, Nome, Idade, Morada)::Inv, LInv2),
  testa(LInv1),
  testa(LInv2),
  assert((excecao(utente(Id,N,I,M)) :-
          utente(Id,N,I,Morada))),
  assert(utente(IdUt, Nome, Idade, Morada)),
  assert(incerto_morada(utente(IdUt,Morada))).
```

Figura 13 - Evolução Incerta Sobre Morada de Utente.

Em termos de evolução de conhecimento interdito, existe um predicado associado a cada um dos atributos interditos (semelhante ao conhecimento incerto), sendo que o primeiro passo é testar os invariantes associados a este tipo de inserção. Caso tudo corra bem, é inserida uma referência sobre o valor nulo a adicionar na base de conhecimento, uma exceção relativa a este mesmo valor nulo, um invariante para não permitir a inserção de conhecimento sobre o registo em questão e, finalmente, é efetuada a inserção sobre o registo. No caso da Figura 14, retrata-se a evolução de uma consulta com o preço interdito.

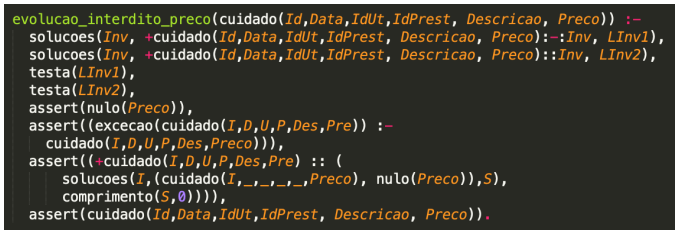

Figura 14 - Evolução Interdita Sobre Preço de Cuidado.

Já no que diz respeito à evolução de conhecimento perfeito negativo, como podem existir situações como "não se conhece a idade do utente 3, Sr. João Emanuel, apenas se sabe que esta está entre 50 e 60 anos, mas não é 56", pode existir conhecimento impreciso/incerto em simultâneo com perfeito negativo, daí não serem removidas as referências a este conhecimento imperfeito aquando da inserção deste novo registo, como se vê na Figura 15.

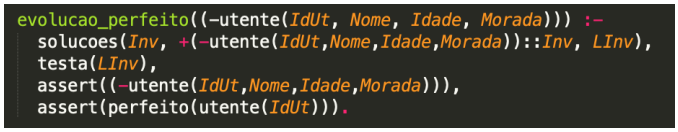

Figura 15 - Evolução Perfeita Negativa.

Por fim, em termos de involução a lógica de pensamento é semelhante, apenas diferindo no que será removido da base de conhecimento, estando sempre tudo em conformidade com os invariantes de remoção. A título de exmplo tenha-se a Figura 16, onde é efetuada a involução de conhecimento interdito relativo à descrição de um cuidado. Começa-se por confirmar que o cuidado em questão existe, tal como o valor nulo a si associado. Em seguida é garantido que a ação está em coformidade com os invariantes definidos. Em caso positivo, o valor nulo é removido da base de conhecimento, tal como a exceção a si associada, o invariante e o cuidado em questão.

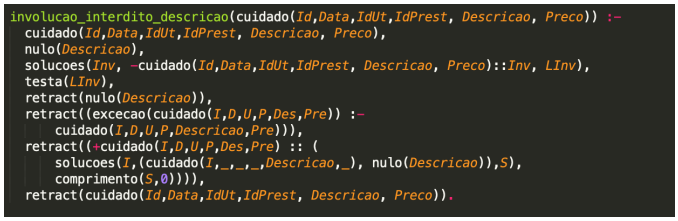

Figura 16 - Involução de Conhecimento Interdito.

3.6 Sistema de Inferência

O desenvolvimento de um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes ao caso em estudo é bastante importante, tendo em conta que este sistema pode atuar como um interpretador de questões. Quer isto dizer que, mediante as questões apresentadas, deve ser devolvida uma conclusão, podendo ser esta verdadeira, falsa ou desconhecida.

O princípio utilizado neste sistema foi o Princípio do Mundo Fechado, ou seja, todo o conhecimento que não está representado, positivo ou negativo, é falso.

O funcionamento do interpretador desenvolvido é bastante simples: dada uma questão este determina a sua veracidade conforme o conhecimento presente na base de conhecimento. Assim, a resposta será verdadeira se a questão estiver de acordo com o conhecimento representado, falsa se existir uma negação explícita à questão e desconhecida caso não existam informações verdadeiras ou negações fortes da questão.

De modo a aperfeiçoar o sistema de inferência, foram desenvolvidos dois predicados para permitir questionar o sistema sobre múltiplas questões em simultâneo: disjunção e conjunção. A ideia por trás destes predicados é bastante semelhante à sua versão na lógica tradicional, mudando apenas o facto de terem em conta valores desconhecidos. Assim, é possível obter respostas sobre mais que uma questão, tornando o sistema mais intuitivo (Figura 17).

```
disjuncao( verdadeiro, X, verdadeiro ).
disjuncao( X, verdadeiro, verdadeiro ).
disjuncao( desconhecido, Y, desconhecido ) :- Y \= verdadeiro.
disjuncao( Y, desconhecido, desconhecido ) :- Y \= verdadeiro.
disjuncao( falso, falso, falso ).

conjuncao( verdadeiro, verdadeiro, verdadeiro ).
conjuncao( falso, _, falso ).
conjuncao( _, falso, falso ).
conjuncao( desconhecido, verdadeiro, desconhecido ).
conjuncao( verdadeiro, desconhecido, desconhecido ).


%------------------------------ - - - - - - - - - - - -  -  -  -  -  -
% Extensao do meta-predicado si: Questao,Resposta -> {V,F}

si( P ## X, V ) :-
  si( P, V1 ),
  si( X, V2 ),
  disjuncao( V1, V2, V ).
si( P && X, V ) :-
  si( P, V1 ),
  si( X, V2 ),
  conjuncao( V1, V2, V ).
si( Questao,verdadeiro ) :-
    Questao.
si( Questao,falso ) :-
    -Questao.
si( Questao,desconhecido ) :-
    nao( Questao ),
    nao( -Questao ).
```

Figura 17 - Sistema de Inferência.

4 Conclusões e Sugestões

Apesar dos conhecimentos adquiridos durante a realização do primeiro projeto prático, este mostrou-se mais complexo, na medida em que obrigou o grupo a desenvolver predicados de evolução e involução em grandes quantidades. Isto fez com que fosse necessário um elevado tempo para efetuar testes e garantir que nada estava a correr mal.

Dito isto, a principal dificuldade encontrada foi retratar os tipos de conhecimento imperfeito da melhor forma possível, com auxílio dos predicados necessários e tendo, sempre, em consideração os invariantes definidos. No entanto, após alguma dedicação e esforço, a maior parte das dificuldades foi ultrapassada, apesar de ainda existir aspetos que podiam ser melhorados no futuro, tal como o facto de apenas ser permitido uma única referência de conhecimento perfeito negativo por registo, ou possuir valores incertos/interditos previamente definidos, simplificando os processos de evolução e involução.

Assim, a realização deste exercício foi bastante importante para compreender na totalidade das diferenças nos vários tipos de conhecimento, bem como aumentar as capacidades de realização de tarefas com a linguagem de programação PROLOG.

Em suma, numa perspetiva geral o grupo está satisfeito com o trabalho desenvolvido e espera conseguir garantir um nível de qualidade elevado no projeto final da unidade curricular.

5 Anexos

Anexo I - Demonstração de Predicados

```
| ?- listing(utente).
utente(1, joao_pimentel, 20, esposende).
utente(2, bruno_veloso, 22, ponte_da_barca).
utente(3, jaime_leite, 20, felgueiras).
utente(4, pedro_goncalves, 20, felgueiras).
utente(5, rodolfo_silva, 20, trofa).
utente(6, joana_amorim, 22, viana_do_castelo).
utente(7, gloria_borga, 53, portimao).
utente(8, andreia_silva, 27, trofa).
utente(18, maria_de_lurdes, 68, nulo2).
utente(16, jose_manuel, nulo5, braga).
utente(15, renato_sancho, nulo4, moinhos).
utente(17, roberto_leque, 32, nulo6).

yes
| ?- evolucao_perfeito(utente(19,joao_jose,33,guarda)).
yes
| ?- listing(utente).
utente(1, joao_pimentel, 20, esposende).
utente(2, bruno_veloso, 22, ponte_da_barca).
utente(3, jaime_leite, 20, felgueiras).
utente(4, pedro_goncalves, 20, felgueiras).
utente(5, rodolfo_silva, 20, trofa).
utente(6, joana_amorim, 22, viana_do_castelo).
utente(7, gloria_borga, 53, portimao).
utente(8, andreia_silva, 27, trofa).
utente(18, maria_de_lurdes, 68, nulo2).
utente(16, jose_manuel, nulo5, braga).
utente(15, renato_sancho, nulo4, moinhos).
utente(17, roberto_leque, 32, nulo6).
utente(19, joao_jose, 33, guarda).

yes
| ?- involucao_perfeito(utente(19,joao_jose,33,guarda)).
yes
| ?- listing(utente).
utente(1, joao_pimentel, 20, esposende).
utente(2, bruno_veloso, 22, ponte_da_barca).
utente(3, jaime_leite, 20, felgueiras).
utente(4, pedro_goncalves, 20, felgueiras).
utente(5, rodolfo_silva, 20, trofa).
utente(6, joana_amorim, 22, viana_do_castelo).
utente(7, gloria_borga, 53, portimao).
utente(8, andreia_silva, 27, trofa).
utente(18, maria_de_lurdes, 68, nulo2).
utente(16, jose_manuel, nulo5, braga).
utente(15, renato_sancho, nulo4, moinhos).
utente(17, roberto_leque, 32, nulo6).

yes
| ?-
```

Figura 18 - Teste Evolução e Involução Perfeito.

```
utente(4, pedro_goncalves, 20, felgueiras).
utente(5, rodolfo_silva, 20, trofa).
utente(6, joana_amorim, 22, viana_do_castelo).
utente(7, gloria_borga, 53, portinao).
utente(8, andreia_silva, 27, trofa).
utente(18, maria_de_lurdes, 68, nulo2).
utente(16, jose_manuel, nulo5, braga).
utente(15, renato_sancho, nulo4, moinhos).
utente(17, roberto_leque, 32, nulo6).

yes
| ?- evolucao_incerto_idade(utente(19,rosa_maria,nulo15,matosinhos)).
yes
| ?- listing(utente).
utente(1, joao_pimentel, 20, esposende).
utente(2, bruno_veloso, 22, ponte_da_barca).
utente(3, jaime_leite, 20, felgueiras).
utente(4, pedro_goncalves, 20, felgueiras).
utente(5, rodolfo_silva, 20, trofa).
utente(6, joana_amorim, 22, viana_do_castelo).
utente(7, gloria_borga, 53, portinao).
utente(8, andreia_silva, 27, trofa).
utente(18, maria_de_lurdes, 68, nulo2).
utente(16, jose_manuel, nulo5, braga).
utente(15, renato_sancho, nulo4, moinhos).
utente(17, roberto_leque, 32, nulo6).
utente(19, rosa_maria, nulo15, matosinhos).

yes
| ?- listing(incerto_idade).
incerto_idade(utente(16,nulo5)).
incerto_idade(utente(19,nulo15)).

yes
| ?- listing(excecao).
excecao(cuidado(A,B,C,D,_)) :-
        cuidado(A, B, C, D, nulo1).
excecao(utente(A,B,C,_)) :-
        utente(A, B, C, nulo2).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo5, C).
excecao(prestador(A,B,_,C,D)) :-
        prestador(A, B, nulo7, C, D).
excecao(utente(14,alfredo,74,braga)).
excecao(utente(14,alfredo,74,barca)).
excecao(cuidado(9,31-12-2019,7,6,eletrocardiograma,A)) :-
        A>20,
        A<30.
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo4, C).
excecao(utente(A,B,C,_)) :-
        utente(A, B, C, nulo6).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo15, C).

yes
| ?-
```

Figura 19 - Teste Evolução Incerto Idade.

```
utente(8, andreia_silva, 27, trofa).
utente(18, maria_de_lurdes, 68, nulo2).
utente(16, jose_manuel, nulo5, braga).
utente(15, renato_sancho, nulo4, moinhos).
utente(17, roberto_leque, 32, nulo6).
utente(19, rosa_maria, nulo15, matosinhos).

yes
| ?- listing(incerto_idade).
incerto_idade(utente(16,nulo5)).
incerto_idade(utente(19,nulo15)).

yes
| ?- listing(excecao).
excecao(cuidado(A,B,C,D,_)) :-
        cuidado(A, B, C, D, nulo1).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo5, C).
excecao(prestador(A,B,_,C,D)) :-
        prestador(A, B, nulo7, C, D).
excecao(utente(14,alfredo,74,braga)).
excecao(utente(14,alfredo,74,barca)).
excecao(cuidado(9,31-12-2019,7,6,eletrocardiograma,A)) :-
        A>20,
        A<30.
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo4, C).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo15, C).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo15, C).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo15, C).

yes
| ?- involucao_incerto_idade(utente(19,rosa_maria,nulo15,matosinhos)).
yes
| ?- listing(utente).
utente(1, joao_pimentel, 20, esposende).
utente(2, bruno_veloso, 22, ponte_da_barca).
utente(3, jaime_leite, 20, felgueiras).
utente(4, pedro_goncalves, 20, felgueiras).
utente(5, rodolfo_silva, 20, trofa).
utente(6, joana_amorim, 22, viana_do_castelo).
utente(7, gloria_borga, 53, portimao).
utente(8, andreia_silva, 27, trofa).
utente(18, maria_de_lurdes, 68, nulo2).
utente(16, jose_manuel, nulo5, braga).
utente(15, renato_sancho, nulo4, moinhos).
utente(17, roberto_leque, 32, nulo6).

yes
| ?- listing(incerto_idade).
incerto_idade(utente(16,nulo5)).

yes
| ?-
```

Figura 20 - Teste Evolução Incerto Idade: Parte 2.

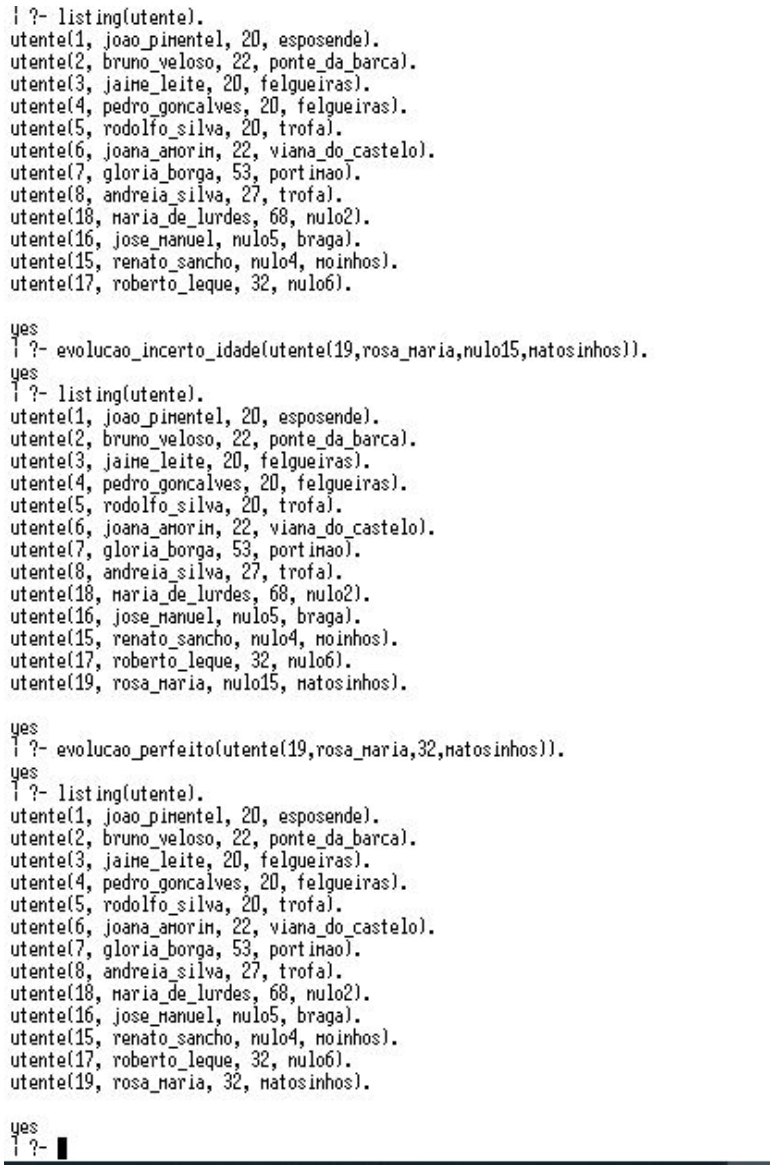

Figura 21 - Teste Sobreposição Conhecimento.

Figura 22 - Teste Involução Interdito Idade.

```
| ?- listing(prestador).
prestador(1, maria_antunes, medicina_geral, trofa_saude, braga).
prestador(2, emanuel_orquidea, cardiologia, cuf, porto).
prestador(3, orlando_beloom, oftalmologia, sao_joao, porto).
prestador(4, jack_pardal, otorrinolarengologia, particular, viana).
prestador(5, antonio_carlos, ginecologia, trofa_saude, braga).
prestador(6, carlos_silva, cirurgia_geral, santo_antonio, porto).
prestador(7, mario_quintas, nulo7, clipovoa, povoa_varzim).
prestador(8, joao_manuel, cardiologia, cuf, porto).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).

yes
| ?- evolucao_interdito_especialidade(prestador(9,gabriela_sousa,nulo19,trofa_saude,trofa)).
yes
| ?- listing(prestador).
prestador(1, maria_antunes, medicina_geral, trofa_saude, braga).
prestador(2, emanuel_orquidea, cardiologia, cuf, porto).
prestador(3, orlando_beloom, oftalmologia, sao_joao, porto).
prestador(4, jack_pardal, otorrinolarengologia, particular, viana).
prestador(5, antonio_carlos, ginecologia, trofa_saude, braga).
prestador(6, carlos_silva, cirurgia_geral, santo_antonio, porto).
prestador(7, mario_quintas, nulo7, clipovoa, povoa_varzim).
prestador(8, joao_manuel, cardiologia, cuf, porto).
prestador(9, gabriela_sousa, nulo19, trofa_saude, trofa).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).
nulo(nulo19).

yes
| ?-
```

Figura 23 - Teste Involução Interdito Especialidade.

```
| ?- listing(prestador).
prestador(1, maria_antunes, medicina_geral, trofa_saude, braga).
prestador(2, emanuel_orquidea, cardiologia, cuf, porto).
prestador(3, orlando_beloom, oftalmologia, sao_joao, porto).
prestador(4, jack_pardal, otorrinolarengologia, particular, viana).
prestador(5, antonio_carlos, ginecologia, trofa_saude, braga).
prestador(6, carlos_silva, cirurgia_geral, santo_antonio, porto).
prestador(7, mario_quintas, nulo7, clipovoa, povoa_varzim).
prestador(8, joao_manuel, cardiologia, cuf, porto).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).

yes
| ?- evolucao_interdito_especialidade(prestador(9,gabriela_sousa,nulo19,trofa_saude,trofa)).
yes
| ?- listing(prestador).
prestador(1, maria_antunes, medicina_geral, trofa_saude, braga).
prestador(2, emanuel_orquidea, cardiologia, cuf, porto).
prestador(3, orlando_beloom, oftalmologia, sao_joao, porto).
prestador(4, jack_pardal, otorrinolarengologia, particular, viana).
prestador(5, antonio_carlos, ginecologia, trofa_saude, braga).
prestador(6, carlos_silva, cirurgia_geral, santo_antonio, porto).
prestador(7, mario_quintas, nulo7, clipovoa, povoa_varzim).
prestador(8, joao_manuel, cardiologia, cuf, porto).
prestador(9, gabriela_sousa, nulo19, trofa_saude, trofa).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).
nulo(nulo19).

yes
| ?- evolucao_perfeito(prestador(9,gabriela_sousa,ortopedia,trofa_saude,trofa)).
no
| ?- evolucao_incerto_especialidade(prestador(9,gabriela_sousa,nulo16,trofa_saude,trofa)).
no
| ?-
```

Figura 24 - Teste Sobreposição Conhecimento Interdito.

```
excecao(prestador(A,B,_,C,D)) :-
        prestador(A, B, nulo7, C, D).
excecao(utente(14,alfredo,74,braga)).
excecao(utente(14,alfredo,74,barca)).
excecao(cuidado(9,31-12-2019,7,6,eletrocardiograma,A)) :-
        A>20,
        A<30.
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo4, C).
excecao(utente(A,B,C,_)) :-
        utente(A, B, C, nulo6).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo16, C).
excecao(prestador(A,B,_,C,D)) :-
        prestador(A, B, nulo19, C, D).

yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).

yes
| ?- evolucao_incerto_preco(cuidado(11,5-5-2016,1,3,visao_desfocada,nulo18)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, nulo18).

yes
| ?- evolucao_perfeito(cuidado(11,5-5-2016,1,3,visao_desfocada,20)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).

yes
| ?-
```

Figura 25 - Teste Sobreposição Perfeito: Cuidados.

```
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).
nulo(nulo19).

yes
| ?- evolucao_interdito_preco(cuidado(12,5-4-2016,1,3,visao_desfocada,nulo18)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).
cuidado(12, 5-4-2016, 1, 3, visao_desfocada, nulo18).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).
nulo(nulo19).
nulo(nulo18).

yes
| ?- evolucao_perfeito(cuidado(12,5-4-2016,1,3,visao_desfocada,15)).
no
| ?- evolucao_incerto_preco(cuidado(12,5-4-2016,1,3,visao_desfocada,nulo20)).
no
| ?-
```

Figura 26 - Teste Sobreposição Interdito: Cuidados.

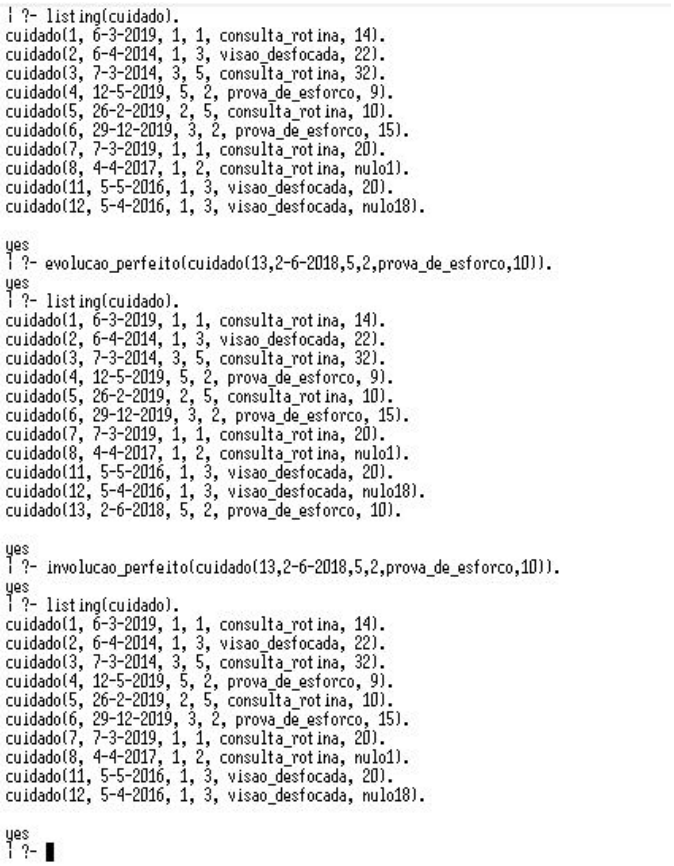

Figura 27 - Teste Evolução e Involução Cuidados.

```
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).
cuidado(12, 5-4-2016, 1, 3, visao_desfocada, nulo18).

yes
| ?- evolucao_incerto_utente(cuidado(13,10-12-2018,nulo12,5,consulta_rotina,20)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).
cuidado(12, 5-4-2016, 1, 3, visao_desfocada, nulo18).
cuidado(13, 10-12-2018, nulo12, 5, consulta_rotina, 20).

yes
| ?- listing(incerto_utente).
incerto_utente(cuidado(13,nulo12)).

yes
| ?- listing(excecao).
excecao(cuidado(A,B,C,D,_)) :-
        cuidado(A, B, C, D, nulo1).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo5, C).
excecao(prestador(A,B,_,C,D)) :-
        prestador(A, B, nulo7, C, D).
excecao(utente(14,alfredo,74,braga)).
excecao(utente(14,alfredo,74,barca)).
excecao(cuidado(9,31-12-2019,7,6,eletrocardiograma,A)) :-
        A>20,
        A<30.
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo4, C).
excecao(utente(A,B,C,_)) :-
        utente(A, B, C, nulo6).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo16, C).
excecao(prestador(A,B,_,C,D)) :-
        prestador(A, B, nulo19, C, D).
excecao(cuidado(A,B,C,D,E,_)) :-
        cuidado(A, B, C, D, E, nulo18).
excecao(cuidado(A,B,_,C,D,E)) :-
        cuidado(A, B, nulo12, C, D, E).
```

Figura 28 - Teste Evolução Incerto Utente.

```
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).
cuidado(12, 5-4-2016, 1, 3, visao_desfocada, nulo18).
cuidado(13, 10-12-2018, nulo12, 5, consulta_rotina, 20).

yes
| ?- listing(incerto_utente).
incerto_utente(cuidado(13,nulo12)).

yes
| ?- listing(excecao).
excecao(cuidado(A,B,C,D,_)) :-
        cuidado(A, B, C, D, nulo1).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo5, C).
excecao(prestador(A,B,_,C,D)) :-
        prestador(A, B, nulo7, C, D).
excecao(utente(14,alfredo,74,braga)).
excecao(utente(14,alfredo,74,barca)).
excecao(cuidado(9,31-12-2019,7,6,eletrocardiograma,A)) :-
        A>20,
        A<30.
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo4, C).
excecao(utente(A,B,C,_)) :-
        utente(A, B, C, nulo6).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo16, C).
excecao(prestador(A,B,_,C,D)) :-
        prestador(A, B, nulo19, C, D).
excecao(cuidado(A,B,C,D,E,_)) :-
        cuidado(A, B, C, D, E, nulo18).
excecao(cuidado(A,B,_,C,D,E)) :-
        cuidado(A, B, nulo12, C, D, E).

yes
| ?- involucao_incerto_utente(cuidado(13,10-12-2018,nulo12,5,consulta_rotina,20)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).
cuidado(12, 5-4-2016, 1, 3, visao_desfocada, nulo18).

yes
| ?- listing(incerto_utente).
yes
```

Figura 29 - .

```
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).
cuidado(12, 5-4-2016, 1, 3, visao_desfocada, nulo18).

yes
| ?- evolucao_incerto_utente(cuidado(13,10-12-2016,nulo33,5,consulta_rotina,20)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).
cuidado(12, 5-4-2016, 1, 3, visao_desfocada, nulo18).
cuidado(13, 10-12-2016, nulo33, 5, consulta_rotina, 20).

yes
| ?- evolucao_perfeito(cuidado(13,10-12-2016,1,5,consulta_rotina,20)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(11, 5-5-2016, 1, 3, visao_desfocada, 20).
cuidado(12, 5-4-2016, 1, 3, visao_desfocada, nulo18).
cuidado(13, 10-12-2016, 1, 5, consulta_rotina, 20).

yes
| ?-
```

Figura 30 - Teste Evolução Incerto Utente: Parte 2.

```
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).

yes
| ?- evolucao_interdito_utente(cuidado(10,5-5-2016,nulo16,2,consulta_rotina,20)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(10, 5-5-2016, nulo16, 2, consulta_rotina, 20).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).
nulo(nulo16).

yes
| ?- evolucao_perfeito(cuidado(10,5-5-2016,5,2,consulta_rotina,20)).
no
| ?- evolucao_incerto_utente(cuidado(10,5-5-2016,nulo10,2,consulta_rotina,20)).
no
| ?-
```

Figura 31 - Teste Evolução Interdito Utente.

```
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(10, 5-5-2016, nulo16, 2, consulta_rotina, 20).

yes
| ?- listing(nulo).
nulo(nulo4).
nulo(nulo6).
nulo(nulo16).

yes
| ?- evolucao_interdito_utente(cuidado(11,5-6-2016,nulo16,2,consulta_rotina,20)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(10, 5-5-2016, nulo16, 2, consulta_rotina, 20).
cuidado(11, 5-6-2016, nulo16, 2, consulta_rotina, 20).

yes
| ?- involucao_interdito_utente(cuidado(11,5-6-2016,nulo16,2,consulta_rotina,20)).
yes
| ?- listing(cuidado).
cuidado(1, 6-3-2019, 1, 1, consulta_rotina, 14).
cuidado(2, 6-4-2014, 1, 3, visao_desfocada, 22).
cuidado(3, 7-3-2014, 3, 5, consulta_rotina, 32).
cuidado(4, 12-5-2019, 5, 2, prova_de_esforco, 9).
cuidado(5, 26-2-2019, 2, 5, consulta_rotina, 10).
cuidado(6, 29-12-2019, 3, 2, prova_de_esforco, 15).
cuidado(7, 7-3-2019, 1, 1, consulta_rotina, 20).
cuidado(8, 4-4-2017, 1, 2, consulta_rotina, nulo1).
cuidado(10, 5-5-2016, nulo16, 2, consulta_rotina, 20).

yes
| ?-
```

Figura 32 - Teste Evolução e Involução Interdito Utente.

```
| ?- si(utente(1,joao_pimentel,20,esposende) && utente(14,alfredo,74,braga),S).
S = desconhecido ?
yes
| ?- si(utente(1,joao_pimentel,20,esposende) && utente(14,alfredo,74,braganca),S).
S = falso ?
yes
| ?- si(utente(1,joao_pimentel,20,esposende) && utente(6,joana_amorim,22,viana_do_castelo),S).
S = verdadeiro ?
yes
```

Figura 33 - Teste Sistema de Inferência: Parte 1.

```
| ?- evolucao_incerto_especialidade(prestador(156,joao_jose,nulo100,cuf,porto)).
yes
| ?- evolucao_perfeito(-prestador(156,joao_jose,cardiologia,cuf,porto)).
yes
| ?- si(prestador(156,joao_jose,dermatologia,cuf,porto),S).
S = desconhecido ?
yes
| ?- si(prestador(156,joao_jose,cardiologia,cuf,porto),S).
S = falso ?
yes
| ?-
```

Figura 34 - Teste Sistema de Inferência: Parte 2.

```
| ?- evolucao_incerto_preco(cuidado(156,2-2-2016,1,2,medicina_geral,nulo100)).
yes
| ?- evolucao_perfeito(-cuidado(156,2-2-2016,1,2,medicina_geral,20)).
yes
| ?- si(cuidado(156,2-2-2016,1,2,medicina_geral,21),S).
S = desconhecido ?
yes
| ?- si(cuidado(156,2-2-2016,1,2,medicina_geral,20),S).
S = falso ?
yes
| ?-
```

Figura 35 - Teste Sistema de Inferência: Parte 3.

```
| ?- evolucao_incerto_utente(cuidado(156,2-2-2016,nulo100,2,medicina_geral,20)).
yes
| ?- evolucao_perfeito(-cuidado(156,2-2-2016,1,2,medicina_geral,20)).
yes
| ?- si(cuidado(156,2-2-2016,3,2,medicina_geral,20),S).
S = desconhecido ?
yes
| ?- si(cuidado(156,2-2-2016,1,2,medicina_geral,20),S).
S = falso ?
yes
| ?-
```

Figura 36 - Teste Sistema de Inferência: Parte 4.