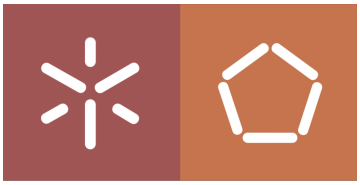

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Trabalho Prático 2

Mestrado Integrado em Engenharia Informática

Sistemas de Representação
de Conhecimento e Raciocínio
(2º semestre, 2020/21)

a83700 **António Santos**
a84912 **Joana A. Gomes**
a84240 **Jorge Vieira**
a85852 **José Magalhães**
a86475 **Paulo R. Pereira**

Braga
Abril 2021

Resumo

O presente relatório é referente à segunda fase do trabalho prático de grupo da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio.

O objetivo é, obviamente, continuar a melhorar as competências na utilização da linguagem de programação em lógica PROLOG, no âmbito da representação de conhecimento e construção de mecanismos de raciocínio para a resolução de problemas.

O objetivo traduz-se em estender o sistema desenvolvido na primeira fase por representar conhecimento positivo e negativo e casos de conhecimento imperfeito.

Serão também manipulados invariantes que designam restrições à inserção e à remoção de conhecimento do sistema, e implementados procedimentos adequados para lidar com a problemática da evolução do conhecimento.

Por fim, apresentar-se-á o sistema de inferência desenvolvido pelo grupo, capaz de implementar todos os mecanismos deste sistema de representação de conhecimento e raciocínio.

Conteúdo

1

Introdução

No âmbito da Unidade Curricular de **Sistemas de Representação de Conhecimento e Raciocínio** foi-nos proposto o desenvolvimento de um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso na área da vacinação global da população portuguesa no contexto COVID que estamos a viver.

A primeira fase do trabalho consistiu em implementar este sistema, capaz de demonstrar certas e determinadas funcionalidades, subjacentes à utilização da linguagem de programação em lógica PROLOG.

O objetivo, agora, é a implementação de funcionalidades que permitam representar conhecimento perfeito positivo e negativo e casos de conhecimento imperfeito, pela utilização de valores nulos dos tipos estudados, manipular invariantes que designem restrições à inserção e remoção do conhecimento, e lidar com a problemática da evolução e involução do conhecimento, desenvolvendo assim um sistema de inferência que implemente os mecanismos necessários.

2

PRELIMINARES

A primeira fase do trabalho habituou-nos a trabalhar com o paradigma da programação em lógica e o motor de inferência PROLOG. Isto permitiu-nos "avançar"com o estudo da programação em lógica, podendo assim representar algumas limitações no nosso universo de discurso.

Estas limitações baseiam-se nos seguintes pressupostos:

- **Pressuposto do Mundo Fechado (PMF)**: toda a informação que não está na base de conhecimento é FALSA;
- **Pressuposto dos Nomes Únicos (PNU)**: duas constantes diferentes não podem designar a mesma entidade, isto é, designam, necessariamente, duas entidades diferentes do universo de discurso;
- **Pressuposto do Domínio Fechado (PDF)**: não existem mais objectos no universo de discurso para além daqueles designados por constantes na base de dados.

Estes pressupostos foram assumidos e salvaguardados na primeira fase do projeto, contudo, tendo em conta que estamos a caracterizar um universo real (vacinação global da população portuguesa no contexto COVID) surgem alguns problemas.

Nas aulas teóricas é dado um exemplo simples que nos ajuda a entender o porquê de na prática surgirem alguns problemas: suponhamos que temos de atravessar uma passagem

de nível. Para isso, temos de ter em atenção se vem ou não um comboio. Ora, em programação em lógica, caso não exista informação sobre um comboio a aproximar-se, o motor de inferência indicar-nos-á que podemos atravessar. Mas isso não prova que não vem nenhum comboio. Prova apenas que não há nenhuma prova que o comboio se aproxima, ou seja, um comboio pode realmente estar a aproximar-se. Na prática, se estivéssemos na passagem de nível, e uma árvore tapasse a nossa visão, então, na nossa "base de conhecimento", não haveria prova de que um comboio se estaria a aproximar, o que não quer dizer que o comboio estivesse realmente a aproximar-se. Apenas não temos conhecimento disso.

Assim, o nosso sistema terá de sofrer algumas alterações. O **PNU** vai continuar a ser salvaguardado, uma vez que não interfere na representação de conhecimento e faz todo o sentido existir no universo a caracterizar, contudo o **PMF** e o **PDF** deixarão de fazer sentido em alguns casos. Serão então substituídos, respetivamente, pelo **Pressuposto do Mundo Aberto (PMA)**, que indica que podem existir factos verdadeiros para além dos representados na base de conhecimento, e pelo **Pressuposto do Domínio Aberto (PDA)**, que indica que podem existir mais objetos do universo de discurso para além daqueles designados pelas constantes da base de conhecimento.

Esta fase do trabalho prático consiste então em **estender** o sistema desenvolvido na primeira fase, de forma a garantir o que foi falado até agora.

A **Programação em Lógica Estendida** permite-nos ainda ter mais um tipo de conclusão para uma dada questão à base de conhecimento. Além da conclusão verdadeira (existe uma prova explícita de que se trata de conhecimento verdadeiro) e falsa (existe uma prova explícita de que se trata de conhecimento falso), temos agora o valor **desconhecido** – não existe informação que permita inferir se é conhecimento verdadeiro ou falso. Isto irá permitir a representação de conhecimento **imperfeito**, que pode ser ainda mais detalhado. É possível identificar os valores mais comuns que podem surgir como informação em falta, ou incompleta, os quais designamos como **valores nulos**. Existem 3 tipos de valores nulos:

- Incerto – não se sabe algum campo, e portanto, é conhecimento desconhecido dentro de um conjunto indeterminado de hipóteses;

- Impreciso – semelhante ao incerto, com a diferença que existe um conjunto determinado de hipóteses para um certo campo, apenas não se sabe qual é o valor correto;
- Interdito – não é permitido conhecer, não há forma de saber.

Por fim, em relação à representação de conhecimento falso, estender a programação em lógica permitir-nos-á representar informação negativa de forma **explícita** (negação forte) e de forma **implícita** (negação por falha na prova).

Assim, teremos uma Extensão à Programação em Lógica completa e bem fundamentada, abrangendo, em suma, as seguintes formas de representação de conhecimento:

- **Conhecimento Perfeito**
  - **Positivo**
  - **Negativo**
    - **Forte**
    - **Por Falha**
- **Conhecimento Imperfeito**
  - **Incerto**
  - **Impreciso**
  - **Interdito**

e, portanto, o nosso objetivo é desenvolver um sistema de inferência capaz de representar, no contexto da vacinação portuguesa contra o COVID, as diferentes formas de conhecimento explicadas.

3

Descrição do Trabalho e Análise de Resultados

Este capítulo está dividido de forma a que o leitor entenda bem a forma como o trabalho está organizado.

Inicialmente, na secção **Base de Conhecimento**, abordamos e descrevemos os predicados que constituem a base de conhecimento do nosso sistema.

A seguir, na secção **Invariantes**, abordamos o controlo da inserção e remoção de conhecimento, apresentando invariantes estruturais e referenciais para esse mesmo controlo.

Nas secções **Evolução do Conhecimento** e **Involução do Conhecimento**, abordamos os procedimentos para os devidos efeitos – adição e remoção de conhecimento, respeitando todos os limites impostos na prática, representados pelos invariantes estudados antes.

Por fim, no capítulo **Sistema de Inferência**, explicamos os motivos da criação de um sistema de inferência, e também a forma como ele foi criado de modo a contemplar todos os mecanismos de raciocínio necessários para o nosso sistema de programação em lógica estendida e representação de conhecimento imperfeito.

No final do documento, encontram-se, em anexo, os ficheiros *.pl* que que constituem toda a nossa Base de Conhecimento.

3.1 Base de Conhecimento

O panorama caracterizado por conhecimento é dado na seguinte forma:

- **utente**: #IdUtente, Número_Segurança_Social, Nome, Género, Data_Nascimento, Email, Telefone, Morada, Profissão, [Doenças_Crónicas], #IdCentroSaúde $\rightsquigarrow \{\mathbb{V}, \mathbb{F}, D\}$
- **centro**: #IdCentroSaúde, Nome, Morada, Telefone, Email $\rightsquigarrow \{\mathbb{V}, \mathbb{F}, D\}$
- **staff**: #IdStaff, IdCentroSaúde, Nome, email $\rightsquigarrow \{\mathbb{V}, \mathbb{F}, D\}$
- **vacinação**: #staf, #utente, Data, Vacina, Toma $\rightsquigarrow \{\mathbb{V}, \mathbb{F}, D\}$
- **fase**: #IdFase, DataInicio, DataFim $\rightsquigarrow \{\mathbb{V}, \mathbb{F}, D\}$

Como se pode reparar, o domínio das soluções foi alterado, de forma a ser possível representar conhecimento imperfeito. A resposta a uma questão à base de conhecimento pode ter então valor verdadeiro ($\mathbb{V}$), falso ($\mathbb{F}$) ou desconhecido (D).

É fácil perceber que predicados devem estar explicitamente contempladas no Pressuposto do Mundo Fechado: **utente**, **staff** e **fase**. Isto significa que qualquer utente, staff ou fase que não esteja na base de conhecimento e que não contenha nenhuma exceção são considerados conhecimento falso.

Esta decisão permite-nos, por exemplo, ao representar conhecimento imperfeito, se não soubermos a idade de um determinado utente, mas soubermos que pertence ao intervalo de valores [45,50], podemos dizer, com certezas, que o utente com idade por exemplo de 55 anos é **falso**, mesmo não tendo esta informação (negação do determinado utente com 55 anos) explícita na nossa base de conhecimento.

Desta forma, definimos os seguintes predicados:

```
-utente(Id, NSS, N, G, DN, E, T, M, P, DC, IdCentro) :-
    nao(utente(Id, NSS, N, G, DN, E, T, M, P, DC, IdCentro)),
    nao(excecao(utente(Id, NSS, N, G, DN, E, T, M, P, DC, IdCentro))).

-staff(Id, IdCentro, N, E) :-
```

```
    nao(staff(Id, IdCentro, N, E)),
    nao(excecao(staff(Id, IdCentro, N, E))).

-vacinacao(IdStaff, IdUtente, D, V, T) :-
    nao(vacinacao(IdStaff, IdUtente, D, V, T)),
    nao(excecao(vacinacao(IdStaff, IdUtente, D, V, T))).

-fase(Id, DI, DF) :-
    nao(fase(Id, DI, DF)),
    nao(excecao(fase(Id, DI, DF))).
```

Para o restante predicado, **centro**, apenas os casos explicitamente negativos são considerados falsos (**PMA**). Isto faz sentido uma vez que nem todos os centros de saúde estão ligados à vacinação contra o COVID.

3.1.1 Conhecimento Perfeito

Conhecimento Perfeito Positivo

Em seguida, é possível ver o "povoamento inicial" do nosso sistema, que corresponde à inserção de factos perfeitos positivos.

```
utente('U0','Romulo Mota','22337840788','M','1936-04-22','RomuloMota@outlook.com',962935227,
    'Rua Francisco Augusto Alvim 4700-004 Braga','Biologo',[ ],'C1').
utente('U1','Goncalo Melo','65105592957','M','1931-09-18','GoncaloMelo1991@hotmail.com',938823950,
    'Travessa 1 de Maio 4700-008 Braga','Geriatra',[ ],'C2').
utente('U5','Apolo Amaral','39446661300','M','1980-05-06','ApoloAmaral@hotmail.com',926158761,
    'Avenida da Liberdade 4760-001 Vila Nova de Famalicao','Reporter',['Insuficiencia hepatica'],'C2').
utente('U7','Afonsino Torres','82144830813','M','1992-12-24','AfonsinoTorres@gmail.com',911799897,
    'Rua Jose Albino Costa e Silva Azurem 4800-004 Guimaraes','Profissional de Saude',[ ],'C2').
utente('U8','Aldo Castro','89885876143','M','1980-04-16','AldoCastro388@hotmail.com',962967073,
    'Rua Doutor Jose Regio Dume 4700-004 Braga','Programador',[ ],'C2').
utente('U9','Tiburcio Guerreiro','08519097436','M','1978-03-16','TiburcioGuerreiro342@outlook.com',
    936810784,'Rua do Cortinhal 4775-001 Cambeses','Militar',['Obesidade'],'C2').
utente('U11','Florisbela Carvalho','38557360298','F','1969-02-01','FlorisbelaCarvalho1969@gmail.com',
    910843987,'Rua Dona Teresa Aldao 4800-004 Guimaraes','Tecnico em edificacoes',[ ],'C4').
utente('U14','Tatiano Carvalho','39247286162','M','1960-12-30','TatianoCarvalho@outlook.com',
    921622230,'Praceta do Viajante Azurem 4800-004 Guimaraes','Biologo',[ ],'C2').
utente('U15','Indro Leal','45346361911','M','1949-07-08','IndroLeal1949@gmail.com',929206797,
```

```
    'Rua de Sao Lourenco da Ordem 4700-004 Braga','Webmaster',[ ],'C1').
utente('U17','Dinarda Matias','29224473940','F','1974-10-09','DinardaMatias1974@outlook.com',
    915914158,'Largo do Rego Dume 4700-008 Braga','DevOps',[ ],'C4').
utente('U18','Marcilene Goncalves','77919813012','M','1989-12-04','MarcileneGoncalves305@hotmail.com',
    924527670,'Rua do Carvalhal Dume 4700-005 Braga','Perito criminal',['Doenca coronaria'],'C1').
utente('U19','Milena Vicente','39280061650','F','2004-06-04','MilenaVicente@gmail.com',969590100,
    'Travessa General Humberto Delgado 4760-001 Vila Nova de Famalicao','Jovem Aprendiz',[ ],'C2').

staff('S0','Leonardina Anjos','12287559265','LeonardinaAnjos49@outlook.com','C2').
staff('S1','Fernando Guerreiro','28567061757','FernandoGuerreiro@outlook.com','C2').
staff('S2','Idelia Paiva','61816783608','IdeliaPaiva@hotmail.com','C1').
staff('S3','Bianca Baptista','55616211371','BiancaBaptista@hotmail.com','C2').
staff('S5','Isandro Azevedo','41601362411','IsandroAzevedo107@gmail.com','C1').
staff('S7','Juliao Abreu','54101439180','JuliaoAbreu@gmail.com','C4').
staff('S10','Juliano Campos','62975266565','JulianoCampos1991@hotmail.com','C2').

centro('C1','Centro de saude de Braga','Rua Paulo Vi 4700-004 Braga','219229219',
    'suporte@debraga.com').
centro('C2','Centro de saude do Caranda','Rua de Goa 4800-004 Guimaraes','246512282',
    'suporte@docaranda.com').
centro('C4','Centro de saude Dr.Paulo Novais','Rua Francisco Ribeiro de Castro 4800-004 Guimaraes',
    '219282688','suporte@drpaulonovais.com').

vacinacao('S10','U1','2021-06-08','Moderna',1).
vacinacao('S10','U1','2021-06-30','Moderna',2).
vacinacao('S15','U2','2021-07-30','Pfizer',1).
vacinacao('S15','U2','2021-08-19','Pfizer',2).
vacinacao('S5','U3','2021-07-07','Pfizer',1).
vacinacao('S5','U3','2021-07-25','Pfizer',2).
vacinacao('S0','U4','2021-04-03','Moderna',1).
vacinacao('S0','U4','2021-04-24','Moderna',2).
vacinacao('S0','U7','2021-06-15','Astrazeneca',1).
vacinacao('S0','U7','2021-07-02','Astrazeneca',2).
vacinacao('S10','U8','2021-07-21','Sputnik V',1).
vacinacao('S10','U8','2021-08-08','Sputnik V',2).
vacinacao('S10','U9','2021-07-05','Pfizer',1).
vacinacao('S10','U9','2021-07-27','Pfizer',2).
vacinacao('S0','U10','2021-07-21','Sputnik V',1).
vacinacao('S0','U10','2021-08-13','Sputnik V',2).
vacinacao('S2','U12','2021-07-16','Moderna',1).
vacinacao('S2','U12','2021-08-05','Moderna',2).
vacinacao('S4','U13','2021-05-11','Astrazeneca',1).
vacinacao('S4','U13','2021-05-30','Astrazeneca',2).
vacinacao('S5','U14','2021-06-30','Sputnik V',1).
vacinacao('S5','U14','2021-07-23','Sputnik V',2).
vacinacao('S3','U17','2021-07-27','Pfizer',1).
```

Nota: Na primeira fase do trabalho, tínhamos um conjunto de factos muito extenso, pelo que decidimos reduzi-lo, uma vez que o que realmente interessa é ter um sistema de inferência capaz de suportar os objetivos pedidos, e os exemplos necessários e suficientes para ver os objetivos a serem cumpridos na prática.

Conhecimento Perfeito Negativo

Como já mencionado, existem duas formas de ter conhecimento negativo: negação forte e negação por falha. Devido ao **PMF**, a forma como os predicados **utente**, **staff** e **fase** estão implementados garante, implicitamente, conhecimento negativo, uma vez que qualquer predicado que não esteja na base de conhecimento com respeito aos mesmos, é conhecimento negativo.

No entanto, decidimos incluir também factos com **negação forte** para estes mesmos predicados. Assim, existe conhecimento negativo com negação forte para todas as "entidades", mas com negação por falha apenas para *utente*, *staff* e *fase*.

Predicados que representam conhecimento perfeito negativo:

```
-utente('U16','Gino Campos','21515788464','M','2004-03-23','GinoCampos@gmail.com',
    920820869,'Lugar da Ordem 4700-008 Braga','Piloto de aviao',[],'C4').

-utente('U21','Rodrigo Matos','90131842525','M','1996-06-04',
    'RodrigoMatos@gmail.com',930344386,'Largo da Confeiteira Dume 4700-005 Braga',
    'Cientista de dados',[],'C1').
```

Os utentes dados como conhecimento perfeito negativo podem corresponder, por exemplo, a utentes que faleceram ou que se recusaram à vacinação.

A mesma lógica foi usada para o *staff*. Os membros do *staff* com IDs *S4* e *S8* podem, por exemplo, ter sido destituídos dos seus cargos no plano de vacinação.

```
-staff('S4','Conceicao Lopes','21645651780','ConceicaoLopes2@outlook.com','C2').

-staff('S8','Dominico Goncalves','65543192708','DominicoGoncalves@hotmail.com',
    'C1').
```

No caso do centro que, por não fazer parte do **PMF**, faz realmente diferença ter conhecimento perfeito negativo, por exemplo, considerando um centro de saúde que tenha fechado (deixado de existir).

```
-centro('C3','Centro de saude do Cavado','Rua de Sao Paulo 4700-004 Braga',
    '233330348','suporte@docavado.com').
```

No caso da vacinação, o conhecimento perfeito negativo é novamente relevante, por exemplo, considerando alguns registos de vacinação que afinal não foram efetuadas, por exemplo, porque o utente não compareceu.

```
-vacinacao('S3','U5','2021-06-08','Sputnik V',1).
-vacinacao('S7','U11','2021-06-19','Pfizer',1).
```

3.1.2 Conhecimento Imperfeito

Como já foi referido em 2, a Extensão à Programação em Lógica permite-nos agora obter mais um tipo de resposta quando questionamos a nossa base de conhecimento. Além do valor verdadeiro e falso, temos agora a hipótese de ter uma resposta com valor **desconhecido**.

As respostas verdadeiras e falsas contemplam o conhecimento perfeito, enquanto que as respostas desconhecidas caracterizam conhecimento **imperfeito**, que será o objeto de estudo desta subsecção do relatório. Foi também referido que existem 3 tipos de conhecimento imperfeito: **incerto**, **impreciso** e **interdito**. Vamos considerar cada um separadamente.

Conhecimento Imperfeito Incerto

Conhecimento Imperfeito Incerto refere-se a predicados dos quais não se sabe algum campo, e portanto, estamos perante conhecimento desconhecido dentro de um conjunto indeterminado ou ilimitado de hipóteses.

- **Utente**

Tendo em conta o nosso caso de estudo, podemos caracterizar como conhecimento imperfeito incerto, por exemplo, utentes cujo ***email*** nos é **desconhecido**. Assim, consideremos o caso de uma utente com *Id* igual a *U13*, de nome *Adelia Amaral*, do qual se sabe o número de telefone, a morada, a profissão e os outros dados, mas se desconhece o *email*, por exemplo por esquecimento de inserção pela utente. Nesta situação, podemos definir como conhecimento incerto, dado que existe um conjunto ilimitado de hipóteses para o *email* do utente.

Para representar este utente, basta colocar, por exemplo *'email_desconhecido'* no campo do *email* e criar um predicado *excecao* que nos permita, quando a base de conhecimento for questionada com o campo *email* com *'email_desconhecido'*, o resultado seja desconhecido.

```
utente('U13','Adelia Amaral','41527730111','F','1944-09-26',email_desconhecido,
    910419566,'Rua Doutor Joao Afonso Almeida Azurem 4800-004 Guimaraes',
    'Bioquimico',[],'C2').
excecao(utente(Id, N, G, DN, E, T, M, P, DC, IdCentro)) :-
    utente(Id, N, G, DN, email_desconhecido, T, M, P, DC, IdCentro)
```

Para exemplificar, questionar se o *email* da Adelia Amaral é 'adeliamaral@gmail.com' daria desconhecido. Mas questionar se a Adelia Amaral é do sexo masculino daria falso! De notar que se o *utente* estivesse assente no PMA, questionar se a Adelia Amaral é do sexo feminino daria desconhecido, visto que isso não está explicitamente negado na base de conhecimento. Podemos ver assim um exemplo que as decisões que tomamos, apresentadas em 2, com respeito aos vários pressupostos, foram boas decisões.

Ainda no que toca ao *utente*, consideramos outras situações em que este tipo de conhecimento seria plausível, que são os casos de **ser desconhecido** o número de **telefone** ou **a morada** do utente.

Só tivemos em conta estes campos dado que, no contexto geral do problema da vacinação, não achamos concebível mais nenhum ser desconhecido, excetuando o *género* que, por poder tomar um conjunto limitado de valores, não se encaixa nesta situação. O *ID*, o *Nome* e o *Número de Utente* são necessários para uma identificação básica do utente. A

data de nascimento é essencial para determinar a fase e que o utente deve ser vacinado, assim como a profissão e as doenças crónicas, sendo o ID do Centro de Saúde também importante no contexto do problema.

Deste modo, nos exemplos seguidos, apresentamos o caso de dois utentes dos quais, por alguma razão, houve um desses campos que não foi preenchido. No caso da utente com *ID* igual a *U4* (Cinderela Nogueira), o campo do número de telefone é desconhecido, e no caso do *U10* (Jansenio Figueiredo) não se conhece a morada.

```
utente('U4','Cinderela Nogueira','50775228700','F','1950-12-08',
    'CinderelaNogueira@outlook.com',tlf_desconhecido,'Rua Paulo Vi 4700-004 Braga',
    'Auxiliar de limpeza',[],'C2').
excecao(utente(Id,N,NSS,G,DN,E,T,M,P,DC,IdCentro)) :-
    utente(Id,N,NSS,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro).

utente('U10','Jansenio Figueiredo','02069412938','M','1983-02-02',
    'JansenioFigueiredo218@outlook.com',914112112,morada_desconhecida,
    'Comissario de bordo',[],'C2').
excecao(utente(Id,N,NSS,G,DN,E,T,M,P,DC,IdCentro)) :-
    utente(Id,N,NSS,G,DN,E,T,morada_desconhecida,P,DC,IdCentro).
```

Nota: Conhecimento assente no PMF, ou seja, os predicados *utente*, *staff* e *fase*, não invalida conhecimento incerto, uma vez que quando a base de conhecimento é questionada com os campos referentes às exceções (por exemplo, *'email_desconhecido'*), a resposta terá valor **desconhecido**.

- **Staff**

No caso do *staff*, acreditamos que para o Conhecimento Imperfeito Incerto apenas deviam ser consideradas situações em que o ***email*** fosse **desconhecido**.

Assim, em conformidade com o já descrito para o *utente*, incluímos a situação de um membro do *staff* (Tatiana Faria) cuja entrada no sistema não tinha *email* associado.

```
staff('S9','Tatiana Faria','59173595792',email_desconhecido,'C4').
excecao(staff(Id,IdC,N,E)) :- staff(Id,IdC,N,email_desconhecido).
```

No que toca às outras entidades (***centro***, ***vacinacao*** e ***fase***), o grupo concluiu que **nenhum dos campos se encaixava em casos de conhecimento imperfeito (no geral)**, dado que no contexto do problema de vacinação saber com certeza esses parâmetros é ou essencial ou é muito relevante.

Conhecimento Imperfeito Impreciso

O conhecimento imperfeito impreciso distingue-se do incerto na medida em que o conjunto de hipóteses para um campo do qual não se sabe o valor é um conjunto limitado ou finito.

- **Utente**

Suponhamos que foi guardado o registo de duas moradas distintas de um utente, e é **desconhecido qual a sua morada** de residência atual, dentro desse **conjunto limitado de duas moradas**. Consideremos o caso de uma utente de nome *Biana Andrade*, com ID *U12*, que se encaixa nesta situação, visto ter associada nos registos as moradas *Rua de Sao Martinho Dume 4700-008 Braga* e *Rua do Monte 4700-009 Braga*.

Para esta situação, criamos exceções para que seja retornado **falso** caso seja questionada uma morada que não seja nenhuma dessas duas, mas se for uma dessas, dará **desconhecido**.

```
excecao(utente('U12','Biana Andrade','03294496759','F','1982-05-02',
    'BianaAndrade@hotmail.com', 932477556,
    'Rua de Sao Martinho Dume 4700-008 Braga','Padeiro',[],'C1')).
excecao(utente('U12','Biana Andrade','03294496759','F','1982-05-02',
    'BianaAndrade@hotmail.com', 932477556,'Rua do Monte 4700-009 Braga',
    'Padeiro',[],'C1')).
```

Este conhecimento imperfeito impreciso poderia ser necessário de implementar nas mais variadas situações, como por exemplo um ***"mix-up"* com dois números de telemóvel**, como é apresentado no exemplo seguinte.

```
excecao(utente('U6','Zelia Abreu','94334540382','F','1995-06-19',
    'ZeliaAbreu@hotmail.com', 966727220,'Rua do Brasil 4775-001 Cambeses',
```

```
    'Eletricista',[],'C2')).
excecao(utente('U6','Zelia Abreu','94334540382','F','1995-06-19',
    'ZeliaAbreu@hotmail.com', 916779510,'Rua do Brasil 4775-001 Cambeses',
    'Eletricista',[],'C2')).
```

Incluimos ainda um exemplo de uma utente com ID *U2*, uma situação em que o *email* não ficou legível, e que portanto fica um **email impreciso** com duas alternativas possíveis,

```
excecao(utente('U2','Bianca Moura','84874182284','F','1958-10-30',
    'BiancaMoura244@gmail.com', 921591637,
    'Rua de Sao Rosendo ( Bispo de Dume ) Dume 4700-008 Braga',
    'Personal trainer',[],'C2')).
excecao(utente('U2','Bianca Moura', '84874182284','F','1958-10-30',
    'BiancaMoura299@gmail.com', 921591637,
    'Rua de Sao Rosendo ( Bispo de Dume ) Dume 4700-008 Braga',
    'Personal trainer',[],'C2')).
```

Ainda no que toca ao *utente*, o grupo decidiu permitir a inserção de utentes com **género desconhecido**, ou seja, utentes que se sabe que são do género masculino ou feminino (um conjunto limitado de duas hipóteses), mas do qual essa informação não é conhecida.

```
excecao(utente('U3','Vilar Nunes','05052988794', 'M','1976-02-17',
    'VilarNunes1976@gmail.com',968452327,'Rua de Remelhe Dume 4700-008 Braga',
    'Vigilante',[],'C1')).
excecao(utente('U3','Vilar Nunes','05052988794','F','1976-02-17',
    'VilarNunes1976@gmail.com',968452327,'Rua de Remelhe Dume 4700-008 Braga',
    'Vigilante',[],'C1')).
```

- **Staff**

No que toca ao *Staff*, em conformidade com o utente, ilustramos um exemplo de um membro do *staff*, de *ID* igual a *S6*, cujo *email* é desconhecido, de entre um conjunto limitado de dois emails.

```
excecao(staff('S6','Almiro Soares','65199252818','AlmiroSoares1991@gmail.com','C2')).
excecao(staff('S6','Almiro Soares','65199252818','AlmiroSoares@gmail.com','C2')).
```

Conhecimento imperfeito impreciso alberga não só casos em que há valores alternativos para um determinado campo, mas também casos em que há um intervalo de valores possíveis. Porém, neste caso de estudo em específico e contexto do problema, o grupo não concebeu possíveis situações em que tal pudesse ser implementado.

É novamente importante referir que, tal como no Conhecimento Imperfeito Incerto, apenas os predicados que fazem parte do **PMF** têm uma diferença significativa, pois daria falso caso a informação não estivesse na base de conhecimento, e assim passa a dar desconhecida, enquanto para os que não fazem parte do **PMF** daria na mesma desconhecido.

Conhecimento Imperfeito Interdito

Conhecimento Imperfeito comporta ainda casos em que o conhecimento é desconhecido e não é permitido conhecê-lo, conhecido como **Conhecimento Imperfeito Interdito**.

- **Utente**

Neste caso de estudo, o grupo implementou conhecimento imperfeito interdito na situação de um utente de ID *U20* e nome *Delmiro Lopes* que não usa tecnologias e que, portanto, torna o seu campo de email impossível de conhecer.

```
utente('U20','Delmiro Lopes','36606379205','M','1956-05-23',email_impossivel,
927819307,'Rua Monte de Baixo 4705-001 Arentim','Cerimonialista',[],'C1').
excecao(utente(Id,N,NSS,G,Dn,E,T,M,P,D,Cs))
    :- utente(Id,N,NSS,G,Dn,email_impossivel,T,M,P,D,Cs).
nulointerdito(email_impossivel).
+utente(Id,N,NSS,G,Dn,E,T,M,P,D,Cs) ::  (solucoes(Id,N,NSS,G,Dn,E,T,M,P,D,Cs),
    utente('U20','Delmiro Lopes','36606379205','M','1956-05-23',email_impossivel,
    927819307,'Rua Monte de Baixo 4705-001 Arentim','Cerimonialista',[],'C1'),
    nao(nulointerdito(email_impossivel))), R),
    comprimento(R,0)).
```

- **Staff**

Embora admitamos a situação mesmo plausível, apresentamos um exemplo equivalente ao

anteriormente apresentado, mas relativo a um membro do staff (de ID *S6*), em que o email é não passível de conhecer.

```
utente('U20','Delmiro Lopes','36606379205','M','1956-05-23',email_impossivel,
927819307,'Rua Monte de Baixo 4705-001 Arentim','Cerimonialista',[],'C1').
excecao(utente(Id,N,NSS,G,Dn,E,T,M,P,D,Cs))
    :- utente(Id,N,NSS,G,Dn,email_impossivel,T,M,P,D,Cs).
nulointerdito(email_impossivel).
+utente(Id,N,NSS,G,Dn,E,T,M,P,D,Cs) ::  (solucoes(Id,N,NSS,G,Dn,E,T,M,P,D,Cs),
    utente('U20','Delmiro Lopes','36606379205','M','1956-05-23',email_impossivel,
    927819307,'Rua Monte de Baixo 4705-001 Arentim','Cerimonialista',[],'C1'),
    nao(nulointerdito(email_impossivel))), R),
    comprimento(R,0)).
```

3.2 Predicados Auxiliares

Na primeira fase do trabalho foram criados alguns predicados que nos auxiliaram no desenvolvimento das várias funcionalidades impostas. Para esta fase, não foram criados mais predicados auxiliares, pelo que é possível rever no anexo A.1 todos os predicados auxiliares usados.

3.3 Invariantes

Adicionar ou remover conhecimento exige procedimentos que controlem essas mesmas ações, isto é, a adição e inserção de conhecimento não deve ser feita de qualquer maneira. Estes procedimentos e o porquê de serem utilizados são devidamente explicados na primeira fase do trabalho prático. Contudo, devido ao facto de agora existir conhecimento perfeito negativo, e também conhecimento imperfeito, sendo agora possível responder com **desconhecido** a certas questões à base de conhecimento, é necessário criar novos invariantes e adaptar alguns já criados.

3.3.1 Invariantes Universais

Alguns invariantes dirão respeito a **todos** os predicados que possam ser inseridos na nossa base de conhecimento, daí os designarmos como invariantes universais.

Assim, em primeiro lugar, uma vez que agora temos conhecimento perfeito positivo e negativo na nossa base de conhecimento, algo que nunca poderia existir seria conhecimento positivo a contradizer conhecimento negativo e vice-versa. Ou seja, se temos X, não pode ser adicionado $-X$, ou, se temos $-X$, não pode ser adicionado X. Isto garante que não existe informação contraditória na nossa base de conhecimento.

```
% Invariantes que não permitem adicionar conhecimento
% perfeito positivo que contradiz conhecimento perfeito negativo
% e vice-versa
+T :: nao(-T).
+(-T) :: nao(T).
```

Em segundo lugar, garantimos também que não existe conhecimento redundante na nossa base de conhecimento por adicionar os seguintes invariantes:

```
% Invariantes que garantem que não existe
% conhecimento perfeito positivo repetido e
% conhecimento perfeito negativo repetido, respetivamente
+T :: (solucoes(T, T, R),
       comprimento(R, 1)).
+(-T) :: (solucoes(T, -T, R),
          comprimento(R, 1)).

% Invariante que garante que não existem exceções repetidas
+(excecao(T)) :: (solucoes(T, excecao(T), R),
                  comprimento(R, 1)).
```

Os restantes invariantes serão abordados nas próximas subsecções separadamente.

Nota: Os invariantes do trabalho anterior diziam respeito a conhecimento perfeito positivo. Como agora temos também conhecimento perfeito negativo é necessário "duplicar" alguns invariantes para conhecimento perfeito negativo.

3.3.2 Invariantes de Utente

A **inserção** de novo conhecimento para o predicado **utente** exige, naturalmente, restrições. Em primeiro lugar, não deve ser possível adicionar um utente com um ID já existente, isto é, o ID de cada utente deve ser único. Além disso, os utentes devem ter no campo referente ao ID do centro, um ID de centro válido. Também, deve ser garantido que utentes com IDs diferentes não tenham exatamente a mesma informação e que o seu género seja 'M' ou 'F'.

```
% Garantir que o ID de cada utente é único:
+utente(Id,_,_,_,_,_,_,_,_,_,_) :: (solucoes(Id, utente(Id,_,_,_,_,_,_,_,_,_,_),
 ↪  R),
                                    comprimento(R, 1)).
+(-utente(Id,_,_,_,_,_,_,_,_,_,_)) :: (solucoes(Id,
 ↪  -utente(Id,_,_,_,_,_,_,_,_,_,_), R),
                                    comprimento(R, 1)).

% Os utentes têm de ter um ID de centro existente
+utente(_,_,_,_,_,_,_,_,_,_,IdC) :: centro(IdC,_,_,_,_).
+(-utente(_,_,_,_,_,_,_,_,_,_,IdC)) :: centro(IdC,_,_,_,_).

% Garantir que utentes com IDs diferentes tem diferente informacao
+utente(Id,_,NSS,_,_,E,TLF,_,_,_,_) :: (solucoes((Id,NSS,E,TLF),
 ↪  utente(_,_,NSS,_,_,E,TLF,_,_,_,_), R),
                                        comprimento(R,1)).
+(-utente(Id,_,NSS,_,_,E,TLF,_,_,_,_)) :: (solucoes((Id,NSS,E,TLF),
 ↪  -utente(_,_,NSS,_,_,E,TLF,_,_,_,_), R),
                                        comprimento(R,1)).

% Garantir que o género do utente é 'M' ou 'F'
+utente(_,_,_,G,_,_,_,_,_,_,_) :: generoValido(G).
+(-utente(_,_,_,G,_,_,_,_,_,_,_)) :: genderValido(G).
```

Por fim, com respeito à **remoção** de conhecimento de utentes, deve ser garantido que não se pode eliminar um utente que tenha vacinações marcadas.

```
-utente(Id,_,_,_,_,_,_,_,_,_,_) :: (findall(Id, vacinacao(_,Id,_,_,_), R),
                                    comprimento(R, 0)).
```

3.3.3 Invariantes de Staff

A **inserção** de novo conhecimento no que diz respeito ao predicado **staff** deverá exigir também algumas restrições. Deve ser garantido que o ID de cada *staff* é único e que funcionários do *staff* com IDs diferentes não tenham a mesma informação.

```
% Garantir que o ID de cada staff é único:
+staff(Id,_,_,_) :: (solucoes(Id, staff(Id,_,_,_), R),
                            comprimento(R, 1)).
+(-staff(Id,_,_,_)) :: (solucoes(Id, -staff(Id,_,_,_), R),
                            comprimento(R, 1)).

%Garantir que funcionarios com IDs diferentes t^em diferente informacao
+staff(_,_,SS,T,C) :: (solucoes((SS,T,C), staff(_,_,SS,T,C), R),
                            comprimento(R,1)).
+(-staff(_,_,SS,T,C)) :: (solucoes((SS,T,C), -staff(_,_,SS,T,C), R),
                            comprimento(R,1)).
```

Com respeito agora à **remoção** de conhecimento, deve-se garantir que não se pode eliminar um funcionário que tenha vacinações marcadas.

```
-staff(Id,_,_,_,_) :: (findall(Id, vacinacao(Id,_,_,_,_), R),
                        comprimento(R, 0)).
```

3.3.4 Invariantes de Centro

Relativamente à **inserção** de conhecimento com respeito ao predicado **centro**, é importante garantir que o ID de cada centro é único e que centros com IDs diferentes têm informação

diferente.

```
% Garantir que o ID de cada centro é único:
+centro(Id,_,_,_,_) :: (solucoes(Id, centro(Id,_,_,_,_), R),
                        comprimento(R, 1)).
+(-centro(Id,_,_,_,_)) :: (solucoes(Id, -centro(Id,_,_,_,_), R),
                        comprimento(R, 1)).

%Garantir que centros com  IDs diferentes têm diferente informacao
+centro(Id,N,_,T,E) :: (solucoes((Id,N,T,E), centro(_,N,_,T,E), R),
                        comprimento(R,1)).
+(-centro(Id,N,_,T,E)) :: (solucoes((Id,N,T,E), -centro(_,N,_,T,E), R),
                        comprimento(R,1)).
```

Com respeito à **remoção** de conhecimento, tal como para o predicado *utente* e para o predicado *staff*, é necessário garantir que não se pode remover um centro que tenha vacinações marcadas.

```
-centro(Id,_,_,_,_) :: (findall(sId,staff(sId,_,_,_,Id),R),
                        comprimento(R, 0)).
```

3.3.5 Invariantes de Vacinação

Em relação às vacinações, o grupo achou por bem que não seria necessário construir invariantes para remover conhecimento sobre este predicado, já que nada impede que uma vacinação seja removida. Contudo, existem, obviamente, algumas limitações aquando da adição de conhecimento.

Em primeiro lugar, adicionar uma vacinação da segunda toma requer que a primeira toma já esteja na base de conhecimento e que ambas as doses sejam do mesmo tipo (houve mudanças nesta política no contexto real do plano de vacinação, mas decidimos não alterar esta decisão uma vez que foi feita na primeira fase do trabalho quando a política real era precisamente essa). Além disso, é necessário que a vacinação seja efetuada num utente existente na base de conhecimento, por um membro do *staff* também existente na base de conhecimento e que trabalhe no centro onde o utente está inscrito.

```
% Adicionar uma vacinaçao da toma 2 requer que a
% toma 1 ja esteja na base de conhecimento
% e que ambas as doses sejam do mesmo tipo
+vacinacao(_,U,_,V,2) :: vacinacao(_,U,_,V,1).
+(-vacinacao(_,U,_,V,2)) :: vacinacao(_,U,_,V,1).

% Garantir que a vacinação é efetuada num utente que esteja na base de
 ↪  conhecimento
+vacinacao(_,U,_,_,_) :: utente(U,_,_,_,_,_,_,_,_,_,_,_).
+(-vacinacao(_,U,_,_,_)) :: utente(U,_,_,_,_,_,_,_,_,_,_,_).

% Garantir que a vacinação é efetuada por um membro da staff que esteja na base de
 ↪  conhecimento
+vacinacao(S,_,_,_,_) :: staff(S,_,_,_,_).
+(-vacinacao(S,_,_,_,_)) :: staff(S,_,_,_,_).

% Garantir que a vacinação é efetuada por um membro da staff
% que trabalhe no centro onde o utente está escrito
+vacinacao(S,U,_,_,_) :: (staff(S,_,_,_,C), utente(U,_,_,_,_,_,_,_,_,_,C)).
+(-vacinacao(S,U,_,_,_)) :: (staff(S,_,_,_,C), utente(U,_,_,_,_,_,_,_,_,_,C)).
```

3.3.6 Invariantes de Fase

Com respeito ao predicado **fase**, consideramos apenas relevante garantir que o inicio de uma fase seja antes do fim da mesma.

```
+fase(_,DI,DF) :: comparaDatasStr(DI,DF).
+(-fase(_,DI,DF)) :: comparaDatasStr(DI,DF).
```

3.4 Evolução de Conhecimento

De forma a lidar com a problemática da evolução do conhecimento, criamos procedimentos relativos aos vários tipos de conhecimento que temos vindo a referir neste relatório (co-

nhecimento perfeito positivo, perfeito negativo, imperfeito incerto, imperfeito impreciso e imperfeito interdito).

3.4.1 Evolução de Conhecimento Perfeito

Na primeira parte do trabalho prático, criamos um meta-predicado *novoConhecimento*, que servia para inserir novo conhecimento na base de conhecimento. Este predicado averiguava se o conhecimento que iríamos adicionar permanecia verdadeiro após ser inserido.

Decidimos nesta segunda parte usar o mesmo meta-predicado para a evolução de conhecimento, abordando tudo o que foi desenvolvido nesta segunda parte do trabalho, e trocamos portanto a designação desse mesmo predicado *novoConhecimento* para ***evolucaoC***, visto ser uma nomenclatura mais intuitiva.

```
evolucaoC(T) :- solucoes(I, +T::I, L),
                insercao(T),
                testaPredicados(L)

solucoes(T,Q,S) :- findall(T,Q,S).

insercao(Q) :- assert(Q).
insercao(Q) :- retract(Q), !, fail.

testaPredicados([]).
testaPredicados([I|L]) :- I, testaPredicados(L).
```

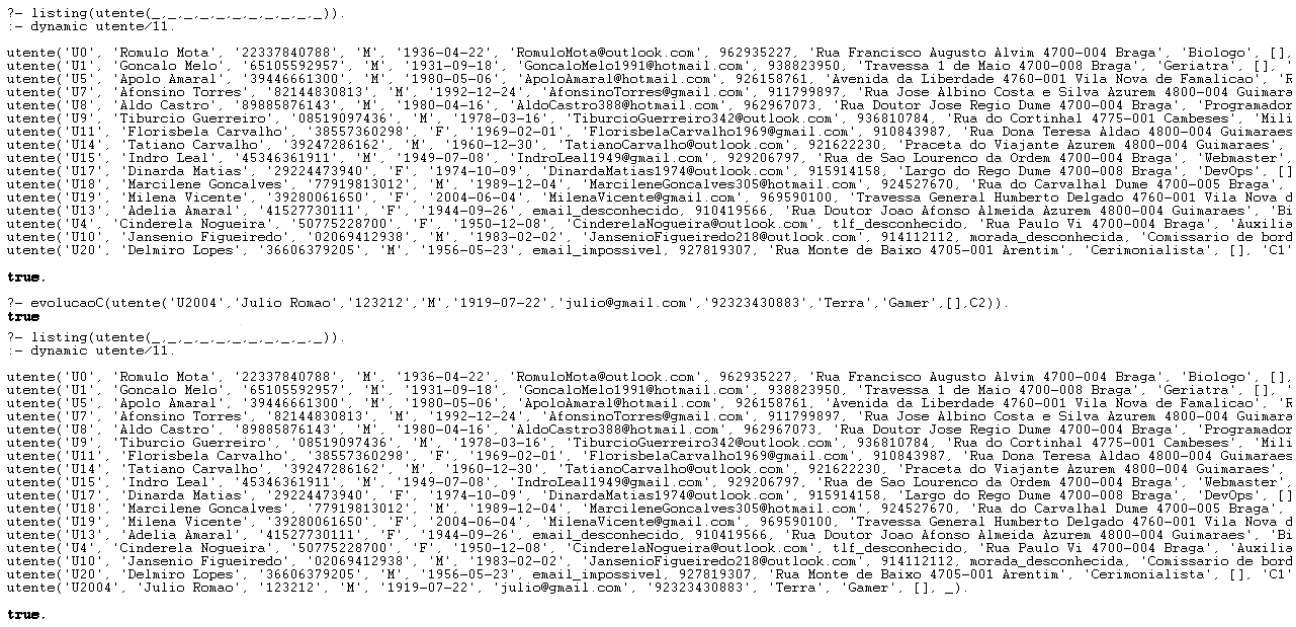

Figura 3.1: Exemplo de evolução de Conhecimento Perfeito

Este meta-predicado *evolucaoC* é muito útil para adicionar conhecimento de qualquer predicado, uma vez que garante que todos os invariantes são respeitados aquando se adiciona conhecimento. Será usado para adicionar conhecimento perfeito positivo, exceções, nulos interditos, entre outros.

Evolução de Conhecimento Perfeito Positivo

Como já referido, para o Conhecimento Perfeito Positivo, é usado o meta-predicado *evolucaoC*. Contudo, para conhecimento perfeito negativo iremos criar uma "especificação" deste meta predicado, como se verá na subsecção seguinte.

Evolução de Conhecimento Perfeito Negativo

Tendo então como base o meta-predicado *evolucaoC*, o grupo desenvolveu um meta-predicado específico para Conhecimento Perfeito Negativo, que testa os invariantes relativos à inserção deste tipo de conhecimento, inserindo termos como negativos fortes.

```
evolucaoC(T,conheNeg) :- solucoes(I, +(-T)::I, L),
                         insercao(-T),
                         testaPredicados(L).
```

3.4.2 Evolução de Conhecimento Imperfeito

Assim como para o Conhecimento Perfeito, nesta fase foram desenvolvidos procedimentos para a evolução de Conhecimento Imperfeito.

Evolução de Conhecimento Imperfeito Incerto

Foi anteriormente referido que, quando se trata de Conhecimento Imperfeito Incerto, adiciona-se um campo chamado, por exemplo, *email_desconhecido* e uma exceção relativa a esse campo. Desta forma, permite-se que seja guardada informação na base de conhecimento caso não se saiba algum dado que não seja extremamente relevante.

Para lidar com a problemática de evolução de conhecimento, criamos um predicado para cada um dos dos tipos diferentes dos campos incertos, quer para o *utente*, quer para o *staff*. Para cada um deles é dado como argumento:

- O **predicado a ser adicionado**;
- O **tipo de predicado** (*utente* ou *staff*, pois só temos este tipo de conhecimento para estas duas entidades);
- O **tipo de conhecimento** (*conheImpInc*, ou seja Conhecimento Imperfeito Incerto);
- Qual o **campo desconhecido** (*email*,*morada*,...).

• **Utente**

```
evolucaoC(utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro),
utente, conheImpInc, email) :-
```

```
    evolucaoC(utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro)),
    insercao((excecao(utente(Id,N,Nu,G,DN,_,T,M,P,DC,IdCentro)) :-
        utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro))).
```

```
evolucaoC(utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro),
utente, conheImpInc, tlf) :-
    evolucaoC(utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro)),
    insercao((excecao(utente(Id,N,Nu,G,DN,E,_,M,P,DC,IdCentro)) :-
        utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro))).
```

```
evolucaoC(utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro),
utente, conheImpInc, morada) :-
    evolucaoC(utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro)),
    insercao((excecao(utente(Id,N,Nu,G,DN,E,T,_,P,DC,IdCentro)) :-
        utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro))).
```

- **Staff**

```
evolucaoC(staff(Id,IdC,N,email_desconhecido), staff, conheImpInc, email) :-
    evolucaoC(staff(Id,IdC,N,email_desconhecido)),
    insercao((excecao(staff(Id,IdC,N,_)) :-
        staff(Id,IdC,N,email_desconhecido))).
```

Figura 3.2: Exemplo de evolução de Conhecimento Imperfeito Incerto (morada desconhecida

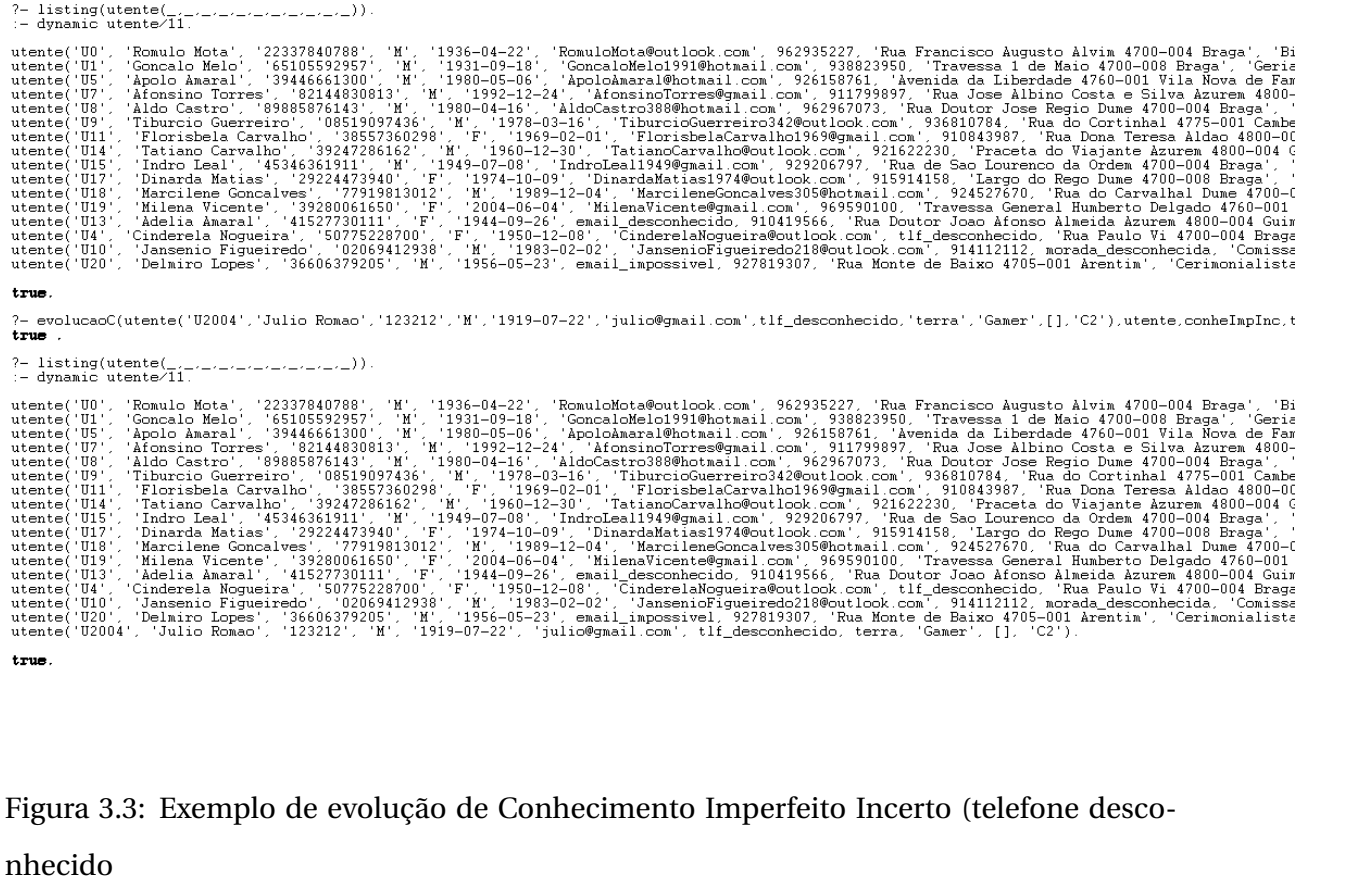

Figura 3.3: Exemplo de evolução de Conhecimento Imperfeito Incerto (telefone desconhecido

Evolução de Conhecimento Imperfeito Impreciso

No caso do Conhecimento Imperfeito Impreciso, como foi referido, são adicionadas exceções para que seja retornado falso caso seja uma opção que não as incluídas nas exceções, e desconhecido caso esteja incluído numa delas.

Desta forma, criamos um predicado em conformidade com o *evolucaoC*, mas, assim como para o Conhecimento Perfeito Negativo, damos-lhe como argumento o tipo de conhecimento (*conheImpImp*, ou seja, Conhecimento Imperfeito Impreciso).

Este predicado verifica se os invariantes que envolvem exceções continuam verdadeiros depois de ser inserido conhecido e, se continuam, então esse conhecimento mantém-se.

```
evolucaoC(T, conheImpImp) :- solucoes(I, +(excecao(T))::I, Lint),
                             insercao(excecao(T)),
                             testaPredicados(Lint).
```

```
?- listing(excecao(_)).
excecao(utente('U12', 'Biana Andrade', '03294496759', 'F', '1982-05-02', 'BianaAndrade@hotmail.com', 932477556, 'Rua de Sao Martinho Dume 4700-008 Braga', 'Padeiro', [], 'C1')).
excecao(utente('U12', 'Biana Andrade', '03294496759', 'F', '1982-05-02', 'BianaAndrade@hotmail.com', 932477556, 'Rua do Monte 4700-009 Braga', 'Padeiro', [], 'C1')).
excecao(utente('U6', 'Zelia Abreu', '94334540382', 'F', '1995-06-19', 'ZeliaAbreu@hotmail.com', 966727220, 'Rua do Brasil 4775-001 Cambeses', 'Eletricista', [], 'C2')).
excecao(utente('U6', 'Zelia Abreu', '94334540382', 'F', '1995-06-19', 'ZeliaAbreu@hotmail.com', 916779510, 'Rua do Brasil 4775-001 Cambeses', 'Eletricista', [], 'C2')).
excecao(utente('U2', 'Bianca Moura', '84874182284', 'F', '1958-10-30', 'BiancaMoura244@gmail.com', 921591637, 'Rua de Sao Rosendo ( Bispo de Dume ) Dume 4700-008 Braga',
excecao(utente('U2', 'Bianca Moura', '84874182284', 'F', '1958-10-30', 'BiancaMoura299@gmail.com', 921591637, 'Rua de Sao Rosendo ( Bispo de Dume ) Dume 4700-008 Braga',
excecao(utente('U3', 'Vilar Nunes', '05052988794', 'M', '1976-02-17', 'VilarNunes1976@gmail.com', 968452327, 'Rua de Remelhe Dume 4700-008 Braga', 'Vigilante', [], 'C1'))
excecao(utente('U3', 'Vilar Nunes', '05052988794', 'F', '1976-02-17', 'VilarNunes1976@gmail.com', 968452327, 'Rua de Remelhe Dume 4700-008 Braga', 'Vigilante', [], 'C1'))
excecao(staff('S6', 'Almiro Soares', '65199252818', 'AlmiroSoares1991@gmail.com', 'C2')).
excecao(staff('S6', 'Almiro Soares', '65199252818', 'AlmiroSoares@gmail.com', 'C2')).
?- evolucaoC(utente('U2004','Julio Romao','123212','M','1919-07-22','julio@gmail.com','92323430883','Terra','Gamer',[],C2),conheImpImp).
true

?- evolucaoC(utente('U2004','Julio Romao','123212','M','1919-07-22','juliogamesPT@gmail.com','92323430883','Terra','Gamer',[],C2),conheImpImp).
true
?- listing(excecao(_)).
excecao(utente('U12', 'Biana Andrade', '03294496759', 'F', '1982-05-02', 'BianaAndrade@hotmail.com', 932477556, 'Rua de Sao Martinho Dume 4700-008 Braga', 'Padeiro', [],
excecao(utente('U12', 'Biana Andrade', '03294496759', 'F', '1982-05-02', 'BianaAndrade@hotmail.com', 932477556, 'Rua do Monte 4700-009 Braga', 'Padeiro', [], 'C1')).
excecao(utente('U6', 'Zelia Abreu', '94334540382', 'F', '1995-06-19', 'ZeliaAbreu@hotmail.com', 966727220, 'Rua do Brasil 4775-001 Cambeses', 'Eletricista', [], 'C2')).
excecao(utente('U6', 'Zelia Abreu', '94334540382', 'F', '1995-06-19', 'ZeliaAbreu@hotmail.com', 916779510, 'Rua do Brasil 4775-001 Cambeses', 'Eletricista', [], 'C2')).
excecao(utente('U2', 'Bianca Moura', '84874182284', 'F', '1958-10-30', 'BiancaMoura244@gmail.com', 921591637, 'Rua de Sao Rosendo ( Bispo de Dume ) Dume 4700-008 Braga',
excecao(utente('U2', 'Bianca Moura', '84874182284', 'F', '1958-10-30', 'BiancaMoura299@gmail.com', 921591637, 'Rua de Sao Rosendo ( Bispo de Dume ) Dume 4700-008 Braga',
excecao(utente('U3', 'Vilar Nunes', '05052988794', 'M', '1976-02-17', 'VilarNunes1976@gmail.com', 968452327, 'Rua de Remelhe Dume 4700-008 Braga', 'Vigilante', [], 'C1'))
excecao(utente('U3', 'Vilar Nunes', '05052988794', 'F', '1976-02-17', 'VilarNunes1976@gmail.com', 968452327, 'Rua de Remelhe Dume 4700-008 Braga', 'Vigilante', [], 'C1'))
excecao(staff('S6', 'Almiro Soares', '65199252818', 'AlmiroSoares1991@gmail.com', 'C2')).
excecao(staff('S6', 'Almiro Soares', '65199252818', 'AlmiroSoares@gmail.com', 'C2')).
excecao(utente('U2004', 'Julio Romao', '123212', 'M', '1919-07-22', 'julio@gmail.com', '92323430883', 'Terra', 'Gamer', [], _)).
excecao(utente('U2004', 'Julio Romao', '123212', 'M', '1919-07-22', 'juliogamesPT@gmail.com', '92323430883', 'Terra', 'Gamer', [], _)).
```

Figura 3.4: Exemplo de evolução de Conhecimento Imperfeito Impreciso

Evolução de Conhecimento Imperfeito Interdito

Em termos de Conhecimento Imperfeito Interdito, temos apenas um exemplo para o *utente* e um exemplo para o *staff*, que cobrem *emails* impossíveis de saber.

- **Utente**

De forma a atingir o objetivo deste predicado, foi necessário criarmos mais um invariante,

que não permite a inserção do *email* interdito *a posteriori*. Apresentamos de seguida esses invariante, e depois o predicado em questão.

```
+utente(Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro) ::
            (solucoes((Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro),
            (utente(Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro), nulointerdito(E)), R),
            comprimento(R,0)).
```

```
evolucaoC(utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro),
    utente, conheImpInt, email) :-
        evolucaoC(utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro)),
        insercao((excecao(utente(Id,N,Nu,G,DN,_,T,M,P,DC,IdCentro)) :-
                utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro))),
        insercao((nulointerdito(email_impossivel))).
```

- **Staff**

Em conformidade com o previamente apresentado no *utente*, procedemos de igual forma para membros de staff cujo *email* é impossível de saber.

```
+staff(Id,IdC,N,E) :: (solucoes((Id,IdC,N,E), (staff(Id,IdC,N,E),
                    nulointerdito(E)), R),
                    comprimento(R,0)).
```

```
evolucaoC(staff(Id,IdC,N,email_impossivel), staff, conheImpInt, email) :-
    evolucaoC(staff(Id,IdC,N,email_impossivel)),
    insercao((excecaoC(staff((Id,IdC,N,_))) :-
                staff(Id,IdC,N,email_impossivel))),
    insercao((nulointerdito(email_impossivel))).
```

3.5 Involução de conhecimento

Foram criados diversos procedimentos relativos aos vários tipos de conhecimento que temos vindo a referir neste relatório (conhecimento perfeito positivo, perfeito negativo, imperfeito

incerto, imperfeito impreciso e imperfeito interdito) para lidar com a involução do conhecimento.

3.5.1 Involução de Conhecimento Perfeito

Na primeira parte do trabalho prático, criamos um meta-predicado *removeConhecimento*, que tinha objetivo de remover conhecimento na base de conhecimento.

Adaptamos este predicado do *TP1* nesta fase do trabalho para o predicado *involucaoC*. Na remoção do conhecimento é averiguado se todos os invariantes estruturais e referenciais que se pretende remover continuam verdadeiros *a posteriori* e só nesse caso é que são removidos.

```
involucaoC(T) :- solucoes(I, -T::I, Linv),
                 remocao(T),
                 testaPredicados(Linv).


remocao(Q) :- retract(Q).
remocao(Q) :- assert(Q), !, fail.
```

Figura 3.5: Exemplo de Involução de Conhecimento Perfeito

Involução de Conhecimento Perfeito Positivo

Para Conhecimento Perfeito Positivo, é usado o meta-predicado *involucaoC*. Para conhecimento perfeito negativo será elaborada uma "especificação" deste meta-predicado, como se verá na subsecção seguinte.

Involução de Conhecimento Perfeito Negativo

Baseando-nos portanto no meta-predicado *involucaoC*, foi desenvolvido um meta-predicado específico para Conhecimento Perfeito Negativo, que testa os invariantes relativos à inserção deste tipo de conhecimento, inserindo termos como negativos fortes.

```
involucaoC(T, conheNeg) :- solucoes(I, -(-T)::I, Linv),
                           remocao(-T),
                           testaPredicados(Linv).
```

3.5.2 Involução de Conhecimento Imperfeito

Assim como para o Conhecimento Perfeito, nesta fase foram desenvolvidos procedimentos para lidar com a problemática da evolução do conhecimento no que toca à remoção do mesmo.

Involução de Conhecimento Imperfeito Incerto

Assim como para a evolução, foram criados predicados que têm como argumento o predicado a ser adicionado, o tipo de predicado, o tipo de conhecimento (*conheImpInc*, Conhecimento Imperfeito Incerto) e qual o campo desconhecido.

Para os utentes, solucionamos a remoção Conhecimento Perfeito Incerto para utente com email desconhecido, telefone desconhecido ou morada desconhecida, e do mesmo modo a remoção desse tipo de conhecimento para membros do *staff* com email desconhecido.

- **Utente**

```
% Remover Conhecimento Perfeito Incerto para utente com email desconhecido
involucaoC(utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro),
utente, conheImpInc, email) :-
    involucaoC(utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro)),
    remocao((excecao(utente(Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro)) :-
        utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro))).

% Remover Conhecimento Perfeito Incerto para utente com telefone desconhecido
involucaoC(utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro),
utente, conheImpInc, tlf) :-
    involucaoC(utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro)),
        remocao((excecao(utente(Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro)) :-
            utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro))).

% Remover Conhecimento Perfeito Incerto para utente com morada desconhecida
```

```
involucaoC(utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro),
utente, conheImpInc, morada) :-
    involucaoC(utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro)),
        remocao((excecao(utente(Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro)) :-
            utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro))).
```

- Staff

```
involucaoC(staff(Id,IdC,N,email_desconhecido), staff, conheImpInc, email) :-
    involucaoC(staff(Id,IdC,N,email_desconhecido)),
    remocao((excecao(staff(Id,IdC,N,E)) :-
        staff(Id,IdC,N,email_desconhecido))).
```

```
?- listing(excecao(_)).
excecao(staff(Id, IdC, N, _)) :-
    staff(Id, IdC, N, email_desconhecido).
excecao(utente('U12', 'Biana Andrade', '03294496759', 'F', '1982-05-02', 'BianaAndrade@hotmail.com', 932477556, 'Rua de Sao Martinho Dume 4700-008
excecao(utente('U12', 'Biana Andrade', '03294496759', 'F', '1982-05-02', 'BianaAndrade@hotmail.com', 932477556, 'Rua do Monte 4700-009 Braga', 'Pa
excecao(utente('U6', 'Zelia Abreu', '94334540382', 'F', '1995-06-19', 'ZeliaAbreu@hotmail.com', 966727220, 'Rua do Brasil 4775-001 Cambeses', 'Ele
excecao(utente('U6', 'Zelia Abreu', '94334540382', 'F', '1995-06-19', 'ZeliaAbreu@hotmail.com', 916779510, 'Rua do Brasil 4775-001 Cambeses', 'Ele
excecao(utente('U2', 'Bianca Moura', '84874182284', 'F', '1958-10-30', 'BiancaMoura244@gmail.com', 921591637, 'Rua de Sao Rosendo ( Bispo de Dume
excecao(utente('U2', 'Bianca Moura', '84874182284', 'F', '1958-10-30', 'BiancaMoura299@gmail.com', 921591637, 'Rua de Sao Rosendo ( Bispo de Dume
excecao(utente('U3', 'Vilar Nunes', '05052988794', 'M', '1976-02-17', 'VilarNunes1976@gmail.com', 968452327, 'Rua de Remelhe Dume 4700-008 Braga',
excecao(utente('U3', 'Vilar Nunes', '05052988794', 'F', '1976-02-17', 'VilarNunes1976@gmail.com', 968452327, 'Rua de Remelhe Dume 4700-008 Braga',
excecao(staff('S6', 'Almiro Soares', '65199252818', 'AlmiroSoares1991@gmail.com', 'C2')).
excecao(staff('S6', 'Almiro Soares', '65199252818', 'AlmiroSoares@gmail.com', 'C2')).

?- involucaoC(utente('U6',_,_,_,_,_,_,_,_,_,_),conheImpImp).
true .

?- listing(excecao(_)).
excecao(utente('U12', 'Biana Andrade', '03294496759', 'F', '1982-05-02', 'BianaAndrade@hotmail.com', 932477556, 'Rua de Sao Martinho Dume 4700-008
excecao(utente('U12', 'Biana Andrade', '03294496759', 'F', '1982-05-02', 'BianaAndrade@hotmail.com', 932477556, 'Rua do Monte 4700-009 Braga', 'Pa
excecao(utente('U6', 'Zelia Abreu', '94334540382', 'F', '1995-06-19', 'ZeliaAbreu@hotmail.com', 916779510, 'Rua do Brasil 4775-001 Cambeses', 'Ele
excecao(utente('U2', 'Bianca Moura', '84874182284', 'F', '1958-10-30', 'BiancaMoura244@gmail.com', 921591637, 'Rua de Sao Rosendo ( Bispo de Dume
excecao(utente('U2', 'Bianca Moura', '84874182284', 'F', '1958-10-30', 'BiancaMoura299@gmail.com', 921591637, 'Rua de Sao Rosendo ( Bispo de Dume
excecao(utente('U3', 'Vilar Nunes', '05052988794', 'M', '1976-02-17', 'VilarNunes1976@gmail.com', 968452327, 'Rua de Remelhe Dume 4700-008 Braga',
excecao(utente('U3', 'Vilar Nunes', '05052988794', 'F', '1976-02-17', 'VilarNunes1976@gmail.com', 968452327, 'Rua de Remelhe Dume 4700-008 Braga',
excecao(staff('S6', 'Almiro Soares', '65199252818', 'AlmiroSoares1991@gmail.com', 'C2')).
excecao(staff('S6', 'Almiro Soares', '65199252818', 'AlmiroSoares@gmail.com', 'C2')).
```

Figura 3.6: Exemplo de Involução de Conhecimento Imperfeito Incerto

Involução de Conhecimento Imperfeito Impreciso

Com a mesma lógica da secção anterior, foi desenvolvido um predicado específico para a remoção de Conhecimento Imperfeito Impreciso.

```
involucaoC(T, conheImpImp) :- solucoes(I, -(excecao(T))::I, Lint),
                              remocao(excecao(T)),
                              testaPredicados(Lint).
```

Involução de Conhecimento Imperfeito Interdito

Em termos de Conhecimento Imperfeito Interdito, assim como para a evolução, temos um exemplo para o *utente* e um exemplo para o *staff*, que cobrem *emails* impossíveis de saber.

- **Utente**

```
involucaoC(utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro),
    utente, conheImpInt, email) :-
        involucaoC(utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro)),
        remocao((excecao(utente(Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro)) :-
            utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro))),
            remocao((nulointerdito(email_impossivel))).
```

- **Staff**

```
involucaoC(staff(Id,IdC,N,email_impossivel), staff, conheImpInt, email) :-
    involucaoC(staff(Id,IdC,N,email_impossivel)),
    remocao((excecaoC(staff((Id,IdC,N,E))) :-
                staff(Id,IdC,N,email_impossivel))),
                remocao((nulointerdito(email_impossivel))).
```

3.6 Sistema de Inferência

A necessidade da criação de um **Sistema de Inferência** deve-se ao facto de o nosso sistema de representação de conhecimento e raciocínio, quando questionado, não responder apenas *verdadeiro* ou *falso*. É agora possível obter resposta do tipo *desconhecido*.

Assim, implementamos o predicado ***si*** (sistema de inferência), que, mediante uma questão *Q*, responde **verdadeiro** se existir uma prova verdadeira na base de conhecimento, **falso** se a questão estiver explicitamente negada (negação forte) na base de conhecimento, e **desconhecido** se não existirem quaisquer provas a respeito de *Q*.

```
si(Q, verdadeiro) :- Q.
si(Q, falso) :- -Q.
si(Q, desconhecido) :- nao(Q),
                       nao(-Q).
```

Uma possível crítica a este sistema de inferência é que ele é muito simples, pois apenas responde a uma questão de cada vez. Assim, decidimos aumentar a complexidade do nosso sistema de inferência por, numa primeira fase, tornar possível responder a mais do que uma questão de cada vez, isto é, responder a uma lista de questões, criando uma lista de respostas.

```
siLista([], []).
siLista([Q|Qs], R) :-
    si(Q, X),
    siLista(Qs, Y),
    R = [X|Y].
```

Decidimos também incluir a hipótese de poder responder a uma conjunção de questões e também a uma disjunção de questões.

Para a conjunção de questões, tendo um conjunto de questões $X_1, X_2, X_3, ..., X_n$, definimos as seguintes três respostas possíveis:

- **verdadeiro** se $\langle \forall i : 1...n : X_i$ é verdadeiro$\rangle$
- **falso** se $\langle \exists i : 1...n : X_i$ é falso$\rangle$
- **desconhecido** se $\langle \exists i : 1...n : X_i$ é desconhecido$\rangle \wedge \langle \forall i : 1...n : X_i$ não é falso$\rangle$

Nota: A notação usada diz respeito ao cálculo de quantificadores de Eindhoven:

1. $\langle \forall x : R : T \rangle$ significa: para qualquer x em R, o termo T tem valor lógico verdadeiro, onde R e T são expressões lógicas que envolvem x

2. $\langle \exists x : R : T \rangle$ significa: para algum x em R, o termo T tem valor lógico verdadeiro, onde R e T são expressões lógicas que envolvem x.

Assim, podemos definir a seguinte tabela de verdade com respeito à conjunção de duas questões:

p	q	$p \wedge q$
verdadeiro	verdadeiro	verdadeiro
falso	_	falso
_	falso	falso
verdadeiro	desconhecido	desconhecido
desconhecido	verdadeiro	desconhecido
desconhecido	desconhecido	desconhecido

cujos predicados em PROLOG correspondem aos seguintes:

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

Desta forma, definimos o predicado *siConjuncao*, que, mediante uma lista de questões, obtém como resposta a conjunção das várias respostas das questões.

```
siConjuncao([],verdadeiro).
siConjuncao([Q|Qs], R) :- si(Q, R1),
                          siConjuncao(Qs,R2),
                          conjuncao(R1,R2,R).
```

Para a disjunção de questões, tendo, de igual forma, um conjunto de questões $X_1, X_2, X_3, ..., X_n$, definimos as seguintes três respostas possíveis:

- **verdadeiro** se $\langle \exists i : 1...n : X_i$ é verdadeiro$\rangle$
- **falso** se $\langle \forall i : 1...n : X_i$ é falso$\rangle$
- **desconhecido** se $\langle \exists i : 1...n : X_i$ é desconhecido$\rangle \wedge \langle \forall i : 1...n : X_i$ não é verdadeiro$\rangle$

Assim, podemos definir a seguinte tabela de verdade com respeito à disjunção de duas questões:

p	q	$p \vee q$
verdadeiro	_	verdadeiro
_	verdadeiro	verdadeiro
falso	falso	falso
falso	desconhecido	desconhecido
desconhecido	falso	desconhecido
desconhecido	desconhecido	desconhecido

cujos predicados em PROLOG correspondem aos seguintes:

```
disjuncao(verdadeiro, _, verdadeiro).
disjuncao(_, verdadeiro, verdadeiro).
disjuncao(falso, falso, falso).
disjuncao(falso, desconhecido, desconhecido).
disjuncao(desconhecido, falso, desconhecido).
disjuncao(desconhecido, desconhecido, desconhecido).
```

Assim, definimos o predicado *siDisjuncao*, que, mediante uma lista de questões, obtém como resposta a disjunção das várias respostas das questões.

```
siDisjuncao([],verdadeiro).
siDisjuncao([Q|Qs], R) :- si(Q, R1),
                          siDisjuncao(Qs,R2),
                          disjuncao(R1,R2,R).
```

4

Conclusões e Sugestões

No fim da segunda fase, o grupo pode dizer que conseguiu desenvolver um sistema de representação de conhecimento e raciocínio capaz de caracterizar corretamente um universo na área da vacinação da população portuguesa face ao COVID-19.

Este sistema, cujas funcionalidades são subjacentes à programação em lógica estendida e à representação de conhecimento imperfeito, inclui diversos tipos de conhecimento – utentes, staff, centro, vacinações e fases.

Foi crucial o estudo dos vários pressupostos, especialmente os pressupostos do Mundo Fechado e do Mundo Aberto, para garantir um sistema próximo da realidade. Desta forma, todas as funcionalidades solicitadas foram corretamente implementadas, permitindo a representação de conhecimento perfeito positivo e negativo e conhecimento imperfeito incerto, impreciso e interdito.

Por fim, a problemática da evolução e involução de conhecimento foi contornada pela construção de invariantes que garantem coerência e consistência da nossa base de conhecimento, para todos os tipos de conhecimento incluídos, e, construiu-se um sistema de inferência complexo capaz de responder de acordo com o novo contradomínio de respostas.

Em suma, este trabalho prático contribuiu para o enriquecimento pedagógico do grupo relativo à programação em lógica estendida usando como motor de inferência o Prolog, no que diz respeito a Sistemas de Representação de Conhecimento e Raciocínio.

Bibliografia

[1] Cesar Analide, Paulo Novais, José Neves. *Sugestões para a Redacção de Relatórios Técnicos*. 2011.

[2] PROLOG,
`https://pt.wikibooks.org/wiki/Prolog`

[3] SWI-Prolog,
`https://www.swi-prolog.org`

A

Código fonte

A.1 auxiliares.pl

```
:- use_module(library(lists)).
:- use_module(library(date)).
:- use_module(library(listing)).

% --------------------------------
% FUNCOES AUXILIARES
% --------------------------------

% Precidado para cacular a idade
calcularIdade(IdU,R) :- utente(IdU,_,_,_,DNasc,_,_,_,_,_,_),
                        split_string(DNasc, "-", "", Sp),
                        nth0(0,Sp,Ano), nth0(1,Sp,Mes), nth0(2,Sp,Dia),
                        get_date_time_value(year, AnoN), get_date_time_value(month,
                         ↪ MesN), get_date_time_value(day, DiaN),
                        atom_number(Ano, Year), atom_number(Mes, Month),
                         ↪ atom_number(Dia, Day),
                        atom_string(AnoN, YearN), atom_number(YearN, YearNN),
                        atom_string(MesN, MonthN), atom_number(MonthN, MonthNN),
                        atom_string(DiaN, DayN), atom_number(DayN, DayNN),
                        Age is (YearNN - Year),
```

```
                        (Month > MonthNN -> AgeN is Age-1 ;
                        Month < MonthNN -> AgeN is Age ;
                            (Day > DayNN -> AgeN is Age-1 ;
                            AgeN is Age)),
                        R = AgeN.

% Obter a data atual
get_date_time_value(Key, Value) :-
    get_time(Stamp),
    stamp_date_time(Stamp, DateTime, local),
    date_time_value(Key, DateTime, Value).

% Predicar para obter a lista de todos os Ids dos utentes
listaIdUtentes(L) :- findall(X,utente(X,_,_,_,_,_,_,_,_,_,_,_),L).

% Intersecao entre duas listas
intersecao([], _, []).
intersecao([H1|T1], L2, [H1|R]) :- member(H1, L2), intersecao(T1, L2, R).
intersecao([_|T1], L2, R) :- intersecao(T1, L2, R).

% Parse de Data
parseData(K,A,M,D) :- split_string(K, "-", "", Sp),
                      nth0(0,Sp,Ano), nth0(1,Sp,Mes), nth0(2,Sp,Dia),
                      atom_number(Ano, Year), atom_number(Mes, Month),
                       ↪  atom_number(Dia, Day),
                      (A is Year, M is Month, D is Day).

% Verifica se data 1 é antes de data 2 (1 = True , 0 = False)
comparaDatas(A1,_,_,A2,_,_,0):- A1 > A2.
comparaDatas(A1,M1,_,A2,M2,_,0):- M1 > M2, A1 >= A2.
comparaDatas(A1,M1,D1,A2,M2,D2,0):- D1 > D2, A1 >= A2, M1 >= M2.
comparaDatas(_,_,_,_,_,_,1).

comparaDatasStr(X1,X2,R):- parseData(X1,A1,M1,D1), parseData(X2,A2,M2,D2),
 ↪  comparaDatas(A1,M1,D1,A2,M2,D2,R).

fase1DataI(DI) :- fase('F1',DI,_).
fase1DataF(DF) :- fase('F1',_,DF).
```

```
fase2DataI(DI) :- fase('F2',DI,_).
fase2DataF(DF) :- fase('F2',_,DF).
fase3DataI(DI) :- fase('F3',DI,_).
fase3DataF(DF) :- fase('F3',_,DF).

% Verifica a que fase pertence um certo dia
checkFase(D,R):- fase1DataI(DI1), comparaDatasStr(D,DI1,A1),
                 fase1DataF(DF1), comparaDatasStr(D,DF1,A2),
                 fase2DataI(DI2), comparaDatasStr(D,DI2,A3),
                 fase2DataF(DF2), comparaDatasStr(D,DF2,A4),
                 fase3DataI(DI3), comparaDatasStr(D,DI3,A5),
                 fase3DataF(DF3), comparaDatasStr(D,DF3,A6),
                 ((A1==0,A2==1) -> R is 1 ;
                 (A3==0,A4==1) -> R is 2 ;
                 (A5==0,A6==1) -> R is 3 ;
                 R is 0).

% Valida o género
generoValido('M').
generoValido('F').

% Comprimento de uma lista
comprimento([],0).
comprimento([_|T],R) :- comprimento(T,N), R is N+1.

% Extensão do metapredicado nao
nao(Questao) :- Questao, !, fail.
nao(_).

% Remoção de conhecimento
remocao(Q) :- retract(Q).
remocao(Q) :- assert(Q), !, fail.

% Procura todas as soluções de uma questão
solucoes(T,Q,S) :- findall(T,Q,S).

% Inserção de conhecimento
insercao(Q) :- assert(Q).
```

```
insercao(Q) :- retract(Q), !, fail.

% Testa todos os predicados numa lista
testaPredicados([]).
testaPredicados([I|L]) :- I, testaPredicados(L).
```

A.2 conhecimento.pl

```
% ---------------------------------
% Conhecimento
% ---------------------------------


% ----- Conhecimento Perfeito -----

% - Conhecimento Perfeito Positivo -

% utente
% #IdUtente, Nome, Nr Utente, Genero, Data Nascimento, Email, Telefone, Morada,
 ↪  Profissao, [Doencas Cronicas], #IdCentroSaude
utente('U0','Romulo
 ↪  Mota','22337840788','M','1936-04-22','RomuloMota@outlook.com',962935227,'Rua
 ↪  Francisco Augusto Alvim 4700-004 Braga','Biologo',[],'C1').
utente('U1','Goncalo
 ↪  Melo','65105592957','M','1931-09-18','GoncaloMelo1991@hotmail.com',938823950,'Travessa
 ↪  1 de Maio 4700-008 Braga','Geriatra',[],'C2').
utente('U5','Apolo
 ↪  Amaral','39446661300','M','1980-05-06','ApoloAmaral@hotmail.com',926158761,'Avenida
 ↪  da Liberdade 4760-001 Vila Nova de Famalicao','Reporter',['Insuficiencia
 ↪  hepatica'],'C2').
utente('U7','Afonsino
 ↪  Torres','82144830813','M','1992-12-24','AfonsinoTorres@gmail.com',911799897,'Rua
 ↪  Jose Albino Costa e Silva Azurem 4800-004 Guimaraes','Profissional de
 ↪  Saude',[],'C2').
```

```
utente('U8','Aldo
 ↪ Castro','89885876143','M','1980-04-16','AldoCastro388@hotmail.com',962967073,'Rua
 ↪ Doutor Jose Regio Dume 4700-004 Braga','Programador',[],'C2').
utente('U9','Tiburcio
 ↪ Guerreiro','08519097436','M','1978-03-16','TiburcioGuerreiro342@outlook.com',936810784,'Rua
 ↪ do Cortinhal 4775-001 Cambeses','Militar',['Obesidade'],'C2').
utente('U11','Florisbela
 ↪ Carvalho','38557360298','F','1969-02-01','FlorisbelaCarvalho1969@gmail.com',910843987,'Rua
 ↪ Dona Teresa Aldao 4800-004 Guimaraes','Tecnico em edificacoes',[],'C4').
utente('U14','Tatiano
 ↪ Carvalho','39247286162','M','1960-12-30','TatianoCarvalho@outlook.com',921622230,'Praceta
 ↪ do Viajante Azurem 4800-004 Guimaraes','Biologo',[],'C2').
utente('U15','Indro
 ↪ Leal','45346361911','M','1949-07-08','IndroLeal1949@gmail.com',929206797,'Rua de
 ↪ Sao Lourenco da Ordem 4700-004 Braga','Webmaster',[],'C1').
utente('U17','Dinarda
 ↪ Matias','29224473940','F','1974-10-09','DinardaMatias1974@outlook.com',915914158,'Largo
 ↪ do Rego Dume 4700-008 Braga','DevOps',[],'C4').
utente('U18','Marcilene
 ↪ Goncalves','77919813012','M','1989-12-04','MarcileneGoncalves305@hotmail.com',924527670,'Rua
 ↪ do Carvalhal Dume 4700-005 Braga','Perito criminal',['Doenca coronaria'],'C1').
utente('U19','Milena
 ↪ Vicente','39280061650','F','2004-06-04','MilenaVicente@gmail.com',969590100,'Travessa
 ↪ General Humberto Delgado 4760-001 Vila Nova de Famalicao','Jovem
 ↪ Aprendiz',[],'C2').

% staff
% #IdStaff, Nome, NSS, Email, IdCentroSaude
staff('S0','Leonardina Anjos','12287559265','LeonardinaAnjos49@outlook.com','C2').
staff('S1','Fernando Guerreiro','28567061757','FernandoGuerreiro@outlook.com','C2').
staff('S2','Idelia Paiva','61816783608','IdeliaPaiva@hotmail.com','C1').
staff('S3','Bianca Baptista','55616211371','BiancaBaptista@hotmail.com','C2').
staff('S7','Juliao Abreu','54101439180','JuliaoAbreu@gmail.com','C4').
staff('S10','Juliano Campos','62975266565','JulianoCampos1991@hotmail.com','C2').

% centro
% #IdCentroSaude, Nome, Morada, Telefone, Email
```

```
centro('C1','Centro de saude de Braga','Rua Paulo Vi 4700-004
 ↪  Braga','219229219','suporte@debraga.com').
centro('C2','Centro de saude do Caranda','Rua de Goa 4800-004
 ↪  Guimaraes','246512282','suporte@docaranda.com').
centro('C4','Centro de saude Dr.Paulo Novais','Rua Francisco Ribeiro de Castro
 ↪  4800-004 Guimaraes','219282688','suporte@drpaulonovais.com').

% vacinacao
%  #IdStaff, #IdUtente, Data, Vacina, Toma
vacinacao('S10','U1','2021-06-08','Moderna',1).
vacinacao('S10','U1','2021-06-30','Moderna',2).
vacinacao('S15','U2','2021-07-30','Pfizer',1).
vacinacao('S15','U2','2021-08-19','Pfizer',2).
vacinacao('S5','U3','2021-07-07','Pfizer',1).
vacinacao('S5','U3','2021-07-25','Pfizer',2).
vacinacao('S0','U4','2021-04-03','Moderna',1).
vacinacao('S0','U4','2021-04-24','Moderna',2).
vacinacao('S0','U7','2021-06-15','Astrazeneca',1).
vacinacao('S0','U7','2021-07-02','Astrazeneca',2).
vacinacao('S10','U8','2021-07-21','Sputnik V',1).
vacinacao('S10','U8','2021-08-08','Sputnik V',2).
vacinacao('S10','U9','2021-07-05','Pfizer',1).
vacinacao('S10','U9','2021-07-27','Pfizer',2).
vacinacao('S0','U10','2021-07-21','Sputnik V',1).
vacinacao('S0','U10','2021-08-13','Sputnik V',2).
vacinacao('S2','U12','2021-07-16','Moderna',1).
vacinacao('S2','U12','2021-08-05','Moderna',2).
vacinacao('S4','U13','2021-05-11','Astrazeneca',1).
vacinacao('S4','U13','2021-05-30','Astrazeneca',2).
vacinacao('S5','U14','2021-06-30','Sputnik V',1).
vacinacao('S5','U14','2021-07-23','Sputnik V',2).
vacinacao('S3','U17','2021-07-27','Pfizer',1).

% fase
% #IdFase, DataInicio, DataFim
fase('F1','2020-12-01','2021-03-31').
fase('F2','2021-04-01','2021-08-31').
fase('F3','2021-09-01','9999-12-31').
```

```
% - Conhecimento Perfeito Negativo -
-utente('U16','Gino
 ↪  Campos','21515788464','M','2004-03-23','GinoCampos@gmail.com',920820869,'Lugar
 ↪  da Ordem 4700-008 Braga','Piloto de aviao',[],'C4').
-utente('U21','Rodrigo
 ↪  Matos','90131842525','M','1996-06-04','RodrigoMatos@gmail.com',930344386,'Largo
 ↪  da Confeiteira Dume 4700-005 Braga','Cientista de dados',[],'C1').

-staff('S4','Conceicao Lopes','21645651780','ConceicaoLopes2@outlook.com','C2').
-staff('S8','Dominico
 ↪  Goncalves','65543192708','DominicoGoncalves@hotmail.com','C1').

-centro('C3','Centro de saude do Cavado','Rua de Sao Paulo 4700-004
 ↪  Braga','233330348','suporte@docavado.com').

-vacinacao('S3','U5','2021-06-08','Sputnik V',1).
-vacinacao('S7','U11','2021-06-19','Pfizer',1).

% ----- Conhecimento Imperfeito -----

% - Conhecimento Imperfeito Incerto -

% UTENTE

% Utente com ID U13 do qual nao se sabe o email
utente('U13','Adelia
 ↪  Amaral','41527730111','F','1944-09-26',email_desconhecido,910419566,'Rua Doutor
 ↪  Joao Afonso Almeida Azurem 4800-004 Guimaraes','Bioquimico',[],'C2').
excecao(utente(Id,N,Nu,G,DN,_,T,M,P,DC,IdCentro)) :-
 ↪  utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro).

% Utente com ID U4 do qual nao se sabe o telefone
utente('U4','Cinderela
 ↪  Nogueira','50775228700','F','1950-12-08','CinderelaNogueira@outlook.com',tlf_desconhecido,'Rua
 ↪  Paulo Vi 4700-004 Braga','Auxiliar de limpeza',[],'C2').
excecao(utente(Id,N,Nu,G,DN,E,_,M,P,DC,IdCentro)) :-
 ↪  utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro).
```

```
% Utente com ID U10 do qual nao se sabe a morada
utente('U10','Jansenio
 ↪  Figueiredo','02069412938','M','1983-02-02','JansenioFigueiredo218@outlook.com',914112112,morad
 ↪  de bordo',[],'C2').
excecao(utente(Id,N,Nu,G,DN,E,T,_,P,DC,IdCentro)) :-
 ↪  utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro).

% STAFF

% Membro do staff com ID S9 do qual nao se sabe a morada
staff('S9','Tatiana Faria','59173595792',email_desconhecido,'C4').
excecao(staff(Id,IdC,N,_)) :- staff(Id,IdC,N,email_desconhecido).

% - Conhecimento Imperfeito Impreciso -

% UTENTE

% Utente com ID U12 com duas moradas possiveis
excecao(utente('U12','Biana
 ↪  Andrade','03294496759','F','1982-05-02','BianaAndrade@hotmail.com',932477556,'Rua
 ↪  de Sao Martinho Dume 4700-008 Braga','Padeiro',[],'C1')).
excecao(utente('U12','Biana
 ↪  Andrade','03294496759','F','1982-05-02','BianaAndrade@hotmail.com',932477556,'Rua
 ↪  do Monte 4700-009 Braga','Padeiro',[],'C1')).

% Utente com ID U6 com dois telefones possiveis
excecao(utente('U6','Zelia
 ↪  Abreu','94334540382','F','1995-06-19','ZeliaAbreu@hotmail.com',966727220,'Rua do
 ↪  Brasil 4775-001 Cambeses','Eletricista',[],'C2')).
excecao(utente('U6','Zelia
 ↪  Abreu','94334540382','F','1995-06-19','ZeliaAbreu@hotmail.com',916779510,'Rua do
 ↪  Brasil 4775-001 Cambeses','Eletricista',[],'C2')).

% Utente com ID U2 com dois emails possiveis
```

```
excecao(utente('U2','Bianca
 ↪  Moura','84874182284','F','1958-10-30','BiancaMoura244@gmail.com',921591637,'Rua
 ↪  de Sao Rosendo ( Bispo de Dume ) Dume 4700-008 Braga','Personal
 ↪  trainer',[],'C2')).
excecao(utente('U2','Bianca
 ↪  Moura','84874182284','F','1958-10-30','BiancaMoura299@gmail.com',921591637,'Rua
 ↪  de Sao Rosendo ( Bispo de Dume ) Dume 4700-008 Braga','Personal
 ↪  trainer',[],'C2')).

% Utente com ID U3 com genero desconhecido (pode ser masculino ou feminino)
excecao(utente('U3','Vilar
 ↪  Nunes','05052988794','M','1976-02-17','VilarNunes1976@gmail.com',968452327,'Rua
 ↪  de Remelhe Dume 4700-008 Braga','Vigilante',[],'C1')).
excecao(utente('U3','Vilar
 ↪  Nunes','05052988794','F','1976-02-17','VilarNunes1976@gmail.com',968452327,'Rua
 ↪  de Remelhe Dume 4700-008 Braga','Vigilante',[],'C1')).

% STAFF
% Membro do staff com ID S6 com dois emails possiveis
excecao(staff('S6','Almiro
 ↪  Soares','65199252818','AlmiroSoares1991@gmail.com','C2')).
excecao(staff('S6','Almiro Soares','65199252818','AlmiroSoares@gmail.com','C2')).

% - Conhecimento Imperfeito Interdito -

% UTENTE

% Utente com ID U20 cujo email e impossivel de saber
utente('U20','Delmiro
 ↪  Lopes','36606379205','M','1956-05-23',email_impossivel,927819307,'Rua Monte de
 ↪  Baixo 4705-001 Arentim','Cerimonialista',[],'C1').
excecao(utente(Id,N,Nu,G,Dn,_,T,M,P,D,Cs)) :-
 ↪  utente(Id,N,Nu,G,Dn,email_impossivel,T,M,P,D,Cs).
nulointerdito(email_impossivel).
```

```
+utente(Id,N,Nu,G,Dn,E,T,M,P,D,Cs) :: (solucoes((Id,N,Nu,G,Dn,E,T,M,P,D,Cs),
 ↪  (utente('U20','Delmiro
 ↪  Lopes','36606379205','M','1956-05-23',email_impossivel,927819307,'Rua Monte de
 ↪  Baixo 4705-001 Arentim','Cerimonialista',[],'C1'),
 ↪  nao(nulointerdito(email_impossivel))), R),
                                        length(R,0)).

% STAFF

% Membro do Staff com ID S5 cujo email e impossivel de saber
staff('S5','Isandro Azevedo','41601362411','IsandroAzevedo107@gmail.com','C1').
excecao(utente(Id,N,Nss,_,Cs)) :- utente(Id,N,Nss,email_impossivel,Cs).
nulointerdito(email_impossivel).
+staff('S5','Isandro Azevedo','41601362411','IsandroAzevedo107@gmail.com','C1') ::
 ↪  (solucoes((_,_,_,_,_), (staff('S5','Isandro
 ↪  Azevedo','41601362411','IsandroAzevedo107@gmail.com','C1'),
 ↪  nao(nulointerdito(email_impossivel))), R),
                                        length(R,0)).
```

A.3 evolucao.pl

```
% ---------------------------------
% Evolucao de Conhecimento
% ---------------------------------

% ----- Evolucao Conhecimento Perfeito -----

% Inserir conhecimento
evolucaoC(T) :- solucoes(I, +T::I, L),
                insercao(T),
                testaPredicados(L).

% ------------------------------------------

% Conhecimento Perfeito Positivo
```

```
% ( usar o evolucaoC )

% ----------------------------------------------

% Conhecimento Perfeito Negativo

% Inserir Conhecimento Perfeito Negativo
evolucaoC(T,conheNeg) :- solucoes(I, +(-T)::I, L),
                         insercao(-T),
                         testaPredicados(L).


% ----- Evolucao do Conhecimento Imperfeito -----

% Evolucao do Conhecimento Imperfeito Incerto

% UTENTE

% Inserir Conhecimento Perfeito Incerto para utente com email desconhecido
evolucaoC(utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro), utente,
 ↪  conheImpInc, email) :-
 ↪  evolucaoC(utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro)),




% Inserir Conhecimento Perfeito Incerto para utente com telefone desconhecido
evolucaoC(utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro), utente,
 ↪  conheImpInc, tlf) :-
 ↪  evolucaoC(utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro)),
```

```
% Inserir Conhecimento Perfeito Incerto para utente com morada desconhecida
evolucaoC(utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro), utente,
↪ conheImpInc, morada) :-
↪ evolucaoC(utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro)),
                                                                    ↪ inserc
                                                                    ↪ :-
                                                                        ↪ ut

% STAFF

% Inserir Conhecimento Perfeito Incerto para membro do staff com email desconhecido
evolucaoC(staff(Id,IdC,N,email_desconhecido), staff, conheImpInc, email) :-
↪ evolucaoC(staff(Id,IdC,N,email_desconhecido)),
                                                          ↪ insercao((excecao(staf
                                                          ↪ :-
                                                              ↪ staff(Id,IdC,N,ema

% ----------------------------------------------

% Evolucao do Conhecimento Imperfeito Impreciso

% Inserir Conhecimento Imperfeito Impreciso

evolucaoC(T, conheImpImp) :- solucoes(I, +(excecao(T))::I, Lint),
                             insercao(excecao(T)),
                             testaPredicados(Lint).

% ----------------------------------------------

% Evolucao do Conhecimento Imperfeito Interdito

% UTENTE
```

```
% Inserir Conhecimento Imperfeito Interdito para utente com email impossivel de
↪ saber
evolucaoC(utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro), utente,
↪ conheImpInt, email) :-
    evolucaoC(utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro)),
    insercao((excecao(utente(Id,N,Nu,G,DN,_,T,M,P,DC,IdCentro)) :-
                utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro))),
    insercao((nulointerdito(email_impossivel))).

% STAFF

% Inserir Conhecimento Imperfeito Interdito para membro do staff com email
↪ impossivel de saber
evolucaoC(staff(Id,IdC,N,email_impossivel), staff, conheImpInt, email) :-
    evolucaoC(staff(Id,IdC,N,email_impossivel)),
    insercao((excecaoC(staff((Id,IdC,N,_))) :-
                staff(Id,IdC,N,email_impossivel))),
    insercao((nulointerdito(email_impossivel))).
```

A.4 invariantes.pl

```
% -----------------------------------------------
% Invariantes
% -----------------------------------------------


% ---------- Invariantes Universais ------------

% Garante que não há conhecimento perfeito positivo repetido
+T :: (solucoes(T, T, R),
       comprimento(R, 1)).

% Garante que não há conhecimento perfeito negativo repetido
+(-T) :: (solucoes(T, -T, R),
       comprimento(R, 1)).
```

```
% Garante não haver contradições entre conhecimento perfeito
+T :: nao(-T).
+(-T) :: nao(T).

% Garante que não há exceções repetidas
+(excecao(T)) :: (solucoes(T,excecao(T),R),
                    comprimento(R,1)).

% --- Invariantes Estruturais e Referenciais ---

% --- Utente

% Garantir que não se pode eliminar um utente que tenha vacinacoes marcadas
-utente(Id,_,_,_,_,_,_,_,_,_,_) :: (findall(Id, vacinacao(_,Id,_,_,_), R),
                                    comprimento(R, 0)).

% Garantir que o ID de cada utente é único:
+utente(Id,_,_,_,_,_,_,_,_,_,_) :: (solucoes(Id, utente(Id,_,_,_,_,_,_,_,_,_,_), R),
                                    comprimento(R, 1)).
+(-utente(Id,_,_,_,_,_,_,_,_,_,_)) :: (solucoes(Id, -utente(Id,_,_,_,_,_,_,_,_,_,_),
 ↪  R),
                                    comprimento(R, 1)).

% Os utentes têm de ter um ID de centro existente
+utente(_,_,_,_,_,_,_,_,_,_,IdC) :: centro(IdC,_,_,_,_).
+(-utente(_,_,_,_,_,_,_,_,_,_,IdC)) :: centro(IdC,_,_,_,_).

% Garantir que utentes com IDs diferentes tem diferente informacao
+utente(Id,_,NSS,_,_,E,TLF,_,_,_,_) :: (solucoes((Id,NSS,E,TLF),
 ↪  utente(_,_,NSS,_,_,E,TLF,_,_,_,_), R),
                                          comprimento(R,1)).
+(-utente(Id,_,NSS,_,_,E,TLF,_,_,_,_)) :: (solucoes((Id,NSS,E,TLF),
 ↪  -utente(_,_,NSS,_,_,E,TLF,_,_,_,_), R),
                                          comprimento(R,1)).

% Garantir que o género do utente é 'M' ou 'F'
+utente(_,_,_,G,_,_,_,_,_,_,_) :: generoValido(G).
+(-utente(_,_,_,G,_,_,_,_,_,_,_)) :: genderValido(G).
```

```
% Impedir a inserção de conhecimento perfeito positivo de utentes com email
 ↪  impossivel de saber - Conhecimento Imperfeito Interdito
+utente(Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro) ::
 ↪  (solucoes((Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro),
 ↪  (utente(Id,N,Nu,G,DN,E,T,M,P,DC,IdCentro), nulointerdito(E)), R),
                                        comprimento(R,0)).


% --- Staff

% Garantir que não posso eliminar um funcionario se tiver vacinacoes marcadas
-staff(Id,_,_,_,_) :: (findall(Id, vacinacao(Id,_,_,_,_), R),
                      comprimento(R, 0)).

% Garantir que o ID de cada staff é único:
+staff(Id,_,_,_) :: (solucoes(Id, staff(Id,_,_,_), R),
                            comprimento(R, 1)).
+(-staff(Id,_,_,_)) :: (solucoes(Id, -staff(Id,_,_,_), R),
                            comprimento(R, 1)).

% Garantir que  funcionarios  com  IDs  diferentes  têm diferente  informacao
+staff(_,_,SS,T,C) :: (solucoes((SS,T,C), staff(_,_,SS,T,C), R),
                      comprimento(R,1)).

+(-staff(_,_,SS,T,C)) :: (solucoes((SS,T,C), -staff(_,_,SS,T,C), R),
                      comprimento(R,1)).

% Impedir a inserção de conhecimento perfeito positivo de membros do staff com email
 ↪  impossivel de saber - Conhecimento Imperfeito Interdito
+staff(Id,IdC,N,E) :: (solucoes((Id,IdC,N,E), (staff(Id,IdC,N,E), nulointerdito(E)),
 ↪  R),
                      comprimento(R,0)).

% Garantir que o staff trabalha num centro existente:
+staff(_,_,_,C) :: centro(C,_,_,_,_).
+(-staff(_,_,_,C)) :: centro(C,_,_,_,_).
```

```
% --- Centro

% Garantir que não posso eliminar um centro se tiver staff a trabalhar
-centro(Id,_,_,_,_) :: (findall(sId,staff(sId,_,_,_,Id),R),
                        comprimento(R, 0)).

% Garantir que não posso eliminar um centro se utentes marcados
-centro(Id,_,_,_,_) :: (findall(UId,utente(UId,_,_,_,_,_,_,_,_,_,_,Id),R),
                        comprimento(R, 0)).

% Garantir que o ID de cada centro é único:
+centro(Id,_,_,_,_) :: (solucoes(Id, centro(Id,_,_,_,_), R),
                        comprimento(R, 1)).
+(-centro(Id,_,_,_,_)) :: (solucoes(Id, -centro(Id,_,_,_,_), R),
                        comprimento(R, 1)).

%Garantir que centros com  IDs diferentes têm diferente informacao
+centro(Id,N,_,T,E) :: (solucoes((Id,N,T,E), centro(_,N,_,T,E), R),
                        comprimento(R,1)).
+(-centro(Id,N,_,T,E)) :: (solucoes((Id,N,T,E), -centro(_,N,_,T,E), R),
                        comprimento(R,1)).


% --- Vacinação

% Adicionar uma vacinaçao da toma 2 requer que a
% toma 1 ja esteja na base de conhecimento
% e que ambas as doses sejam do mesmo tipo
+vacinacao(_,U,_,V,2) :: vacinacao(_,U,_,V,1).
+(-vacinacao(_,U,_,V,2)) :: vacinacao(_,U,_,V,1).

% Garantir que a vacinação é efetuada num utente que esteja na base de conhecimento
+vacinacao(_,U,_,_,_) :: utente(U,_,_,_,_,_,_,_,_,_,_).
+(-vacinacao(_,U,_,_,_)) :: utente(U,_,_,_,_,_,_,_,_,_,_).

% Garantir que a vacinação é efetuada por um membro da staff que esteja na base de
 ↪  conhecimento
+vacinacao(S,_,_,_,_) :: staff(S,_,_,_,_).
```

```
+(-vacinacao(S,_,_,_,_)) :: staff(S,_,_,_,_).

% Garantir que a vacinação é efetuada por um membro da staff
% que trabalhe no centro onde o utente está escrito
+vacinacao(S,U,_,_,_) :: (staff(S,_,_,_,C), utente(U,_,_,_,_,_,_,_,_,_,_,C)).
+(-vacinacao(S,U,_,_,_)) :: (staff(S,_,_,_,C), utente(U,_,_,_,_,_,_,_,_,_,_,C)).

% --- Fase

% Garantir que o inicio de uma fase seja antes do fim da mesma
+fase(_,DI,DF) :: comparaDatasStr(DI,DF).
+(-fase(_,DI,DF)) :: comparaDatasStr(DI,DF).
```

A.5 involucao.pl

```
% ---------------------------------
% Involução de Conhecimento
% ---------------------------------


% ----- Involucao Conhecimento Perfeito -----

% Remover conhecimento
involucaoC(T) :- solucoes(I, -T::I, L),
                 remocao(T),
                 testaPredicados(L).

% ------------------------------------------------

% Involucao do Conhecimento Perfeito Positivo

% ( involucaoC )

% ------------------------------------------------

% Involucao do Conhecimento Perfeito Negativo
```

```
involucaoC(T,conheNeg) :- solucoes(I, -(-T)::I, L),
                          remocao(-T),
                          testaPredicados(L).

% ----- Involucao do Conhecimento Imperfeito -----

% Involucao do Conhecimento Imperfeito Incerto

% UTENTE

% Remover Conhecimento Perfeito Incerto para utente com email desconhecido
involucaoC(utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro), utente,
 ↪  conheImpInc, email) :-
 ↪  involucaoC(utente(Id,N,Nu,G,DN,email_desconhecido,T,M,P,DC,IdCentro)),



% Remover Conhecimento Perfeito Incerto para utente com telefone desconhecido
involucaoC(utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro), utente,
 ↪  conheImpInc, tlf) :-
 ↪  involucaoC(utente(Id,N,Nu,G,DN,E,tlf_desconhecido,M,P,DC,IdCentro)),



% Remover Conhecimento Perfeito Incerto para utente com morada desconhecida
involucaoC(utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro), utente,
 ↪  conheImpInc, morada) :-
 ↪  involucaoC(utente(Id,N,Nu,G,DN,E,T,morada_desconhecida,P,DC,IdCentro)),

                                                                  ↪  remoca
                                                                  ↪  :-
```

```
                                                                    ↪ ut

% STAFF

% Remover Conhecimento Perfeito Incerto para membro do staff com email desconhecido
involucaoC(staff(Id,IdC,N,email_desconhecido), staff, conheImpInc, email) :-
 ↪ involucaoC(staff(Id,IdC,N,email_desconhecido)),

                                                                    ↪ remocao((excecao(staff
                                                                    ↪ :-
                                                                    ↪ staff(Id,IdC,N,email_d


% ----------------------------------------------

% Involucao do Conhecimento Imperfeito Impreciso

% Remover Conhecimento Imperfeito Impreciso

involucaoC(T, conheImpImp) :- solucoes(I, -(excecao(T))::I, Lint),
                              remocao(excecao(T)),
                              testaPredicados(Lint).

% ----------------------------------------------

% Evolucao do Conhecimento Imperfeito Interdito

% UTENTE

% Retirar Conhecimento Imperfeito Interdito para utente com email impossivel de
 ↪ saber
involucaoC(utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro), utente,
 ↪ conheImpInt, email) :-
    involucaoC(utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro)),
    remocao((excecao(utente(Id,N,Nu,G,DN,_,T,M,P,DC,IdCentro)) :-
                utente(Id,N,Nu,G,DN,email_impossivel,T,M,P,DC,IdCentro))),
    remocao((nulointerdito(email_impossivel))).
```

```
% Retirar Conhecimento Imperfeito Interdito para membro do staff com email
 ↪  impossivel de saber
involucaoC(staff(Id,IdC,N,email_impossivel), staff, conheImpInt, email) :-
    involucaoC(staff(Id,IdC,N,email_impossivel)),
    remocao((excecaoC(staff((Id,IdC,N,_))) :-
                staff(Id,IdC,N,email_impossivel))),
    remocao((nulointerdito(email_impossivel))).
```

A.6 tp2.pl

```
% -----------------------------------
% SRCR TP2
% -----------------------------------


% ------ Declarações Iniciais -------

:- set_prolog_flag(discontiguous_warnings,off).
:- set_prolog_flag(single_var_warnings,off).
:- set_prolog_flag(unknown,fail).

:- op(900,xfy,'::').
:- op(1100,xfy,'??').

:- dynamic '-'/1.
:- dynamic utente/11.
:- dynamic staff/5.
:- dynamic centro/5.
:- dynamic vacinacao/5.
:- dynamic fase/3.

:- dynamic excecao/1.
:- dynamic nulointerdito/1.

:- discontiguous '-'/1.
:- discontiguous utente/11.
```

```
:- discontiguous staff/5.
:- discontiguous centro/5.
:- discontiguous vacinacao/5.
:- discontiguous fase/3.

:- discontiguous excecao/1.
:- discontiguous nulointerdito/1.
:- discontiguous (::)/2.
:- discontiguous evolucaoC/2.
:- discontiguous evolucaoC/4.
:- discontiguous involucaoC/2.
:- discontiguous involucaoC/4.

% ------- Carregar predicados -------

:- include('conhecimento.pl').
:- include('evolucao.pl').
:- include('involucao.pl').
:- include('auxiliares.pl').
:- include('invariantes.pl').


% -- PMF para Utente, Staff, Vacinação e Fase --

-utente(Id, NSS, N, G, DN, E, T, M, P, DC, IdCentro) :-
    nao(utente(Id, NSS, N, G, DN, E, T, M, P, DC, IdCentro)),
    nao(excecao(utente(Id, NSS, N, G, DN, E, T, M, P, DC, IdCentro))).

-staff(Id, IdCentro, N, E) :-
    nao(staff(Id, IdCentro, N, E)),
    nao(excecao(staff(Id, IdCentro, N, E))).

-vacinacao(IdStaff, IdUtente, D, V, T) :-
    nao(vacinacao(IdStaff, IdUtente, D, V, T)),
    nao(excecao(vacinacao(IdStaff, IdUtente, D, V, T))).

-fase(Id, DI, DF) :-
    nao(fase(Id, DI, DF)),
```

```
    nao(excecao(fase(Id, DI, DF))).

% ----- Sistema de Inferência ------

% Extensao do meta-predicado si
% que responde a uma única questão
si(Q, verdadeiro) :- Q.
si(Q, falso) :- -Q.
si(Q, desconhecido) :- nao(Q),
                       nao(-Q).

% extensão do meta predicado siLista que dada uma lista de questões
% responde às várias colocando as respostas numa lista
siLista([], []).
siLista([Q|Qs], R) :- si(Q, X),
                      siLista(Qs, Y),
                      R = [X|Y].

% % Extensao do predicado conjuncao
conjuncao(verdadeiro, verdadeiro, verdadeiro).
conjuncao(falso, _, falso).
conjuncao(_, falso, falso).
conjuncao(verdadeiro, desconhecido, desconhecido).
conjuncao(desconhecido, verdadeiro, desconhecido).
conjuncao(desconhecido, desconhecido, desconhecido).

% Extensao do predicado siConjuncao que dado uma lista de questões
% obtém como resposta a conjunção das várias respostas das questões.
siConjuncao([],_).
siConjuncao([Q],R) :- si(Q, R).
siConjuncao([Q|Qs], R) :- si(Q, R1),
                          siConjuncao(Qs,R2),
                          conjuncao(R1,R2,R).

% Extensao do predicado disjuncao
disjuncao(verdadeiro, _, verdadeiro).
disjuncao(_, verdadeiro, verdadeiro).
disjuncao(falso, falso, falso).
```

```
disjuncao(falso, desconhecido, desconhecido).
disjuncao(desconhecido, falso, desconhecido).
disjuncao(desconhecido, desconhecido, desconhecido).

% Extensao do predicado siDisjuncao que dado uma lista de questões
% obtém como resposta a disjunção das várias respostas das questões.
siDisjuncao([],_).
siDisjuncao([Q],R) :- si(Q, R).
siDisjuncao([Q|Qs], R) :- si(Q, R1),
                          siDisjuncao(Qs,R2),
                          disjuncao(R1,R2,R).

% no "exercício" por o código
```