Um simples e incompleto sistema de dedução natural

Um sistema de dedução natural é um sistema composto por um conjunto de regras de derivação que permite verificar se um dado sequente lógico é verdadeiro. As principais regras de dedução natural envovlendo os operadores são apresentadas a seguir:

$$\frac{\alpha \in \Gamma}{\Gamma \vdash \alpha}\ \{ID\}$$

$$\frac{\Gamma \vdash \alpha \quad \Gamma \vdash \beta}{\Gamma \vdash \alpha \wedge \beta}\ \{\wedge_I\}$$

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta}\ \{\wedge_I\}$$

$$\frac{\alpha \wedge \beta}{\alpha}\ \{\wedge_{EE}\} \quad \frac{\alpha \wedge \beta}{\beta}\ \{\wedge_{ED}\}$$

$$\frac{\alpha \to \beta \quad \alpha}{\beta}\ \{\to_E\}$$

$$\frac{\Gamma \vdash \alpha}{\Gamma \vdash \alpha \vee \beta}\ \{\vee_{IE}\} \quad \frac{\Gamma \vdash \beta}{\Gamma \vdash \alpha \vee \beta}\ \{\vee_{ID}\}$$

$$\frac{\Gamma \vdash \alpha \vee \beta \quad \Gamma \cup \{\alpha\} \vdash \gamma \quad \Gamma \cup \{\beta\} \vdash \gamma}{\Gamma \vdash \gamma}\ \{\vee_E\}$$

$$\frac{\Gamma \cup \{\neg\alpha\} \vdash \bot \quad \Gamma \cup \{\beta\} \vdash \gamma}{\Gamma \vdash \alpha}\ \{RAA\}$$

Notes que nem todas as regras de derivação, estão presentes, como por exemplo a regra de contradição e da redução ao abusrdo. Todas as regras coloridas em azul já foram codificadas no arquivo fonte de exemplo fornecido com este enunciado.

Sua tarefa é codificar o sistema de dedução apresentado em Prolog de modo que seja possível provar a corretude de um dado sequente lógico.

Optou-se por codificar o sistema de dedução natural pela coidificação de contexto de prova denotado pelo functor `ctx(H,O)`, onde `H` é uma lista de fórmulas lógicas (o conunto de hipóteses) e `O` é uma formula objetivo. Cada oeprador da lógica proposicional pode ser representado por um functor. E fórmula pode ser descrita como uma construção de funtors. O seguinte sequente $A \to B, A \vee B \vdash B$ pode ser representado pelo functor `ctx([imp(a,b),and(a,b)],b)`. Para maior conveniência o operador de negação pode ser implementado em termos da implicação usando a equivalênica $\neg A \equiv A \to \bot$

As regras de dedução natural propriamente ditas são representadas por regras codificadas diretamente em prolog. Estas regras podem ser divididas em duas categorias: deduce e proof.

A regra deduce sucede se uma dada formula pode ser obtida do contexto e compreende as regras de eliminação de operadores.

```
deduce(ctx(H,A),['id']).
deduce(ctx(H,A),['and-ee']) :- member(and(A,_),H).
deduce(ctx(H,B),['and-ed']) :- member(and(_,B),H).
deduce(ctx(H,B),['imp-e','id'|R]) :- member(imp(A,B),H), deduce(ctx(H,A),R).
```

A regra proof tentar construir uma formula objetivo a partir do contexto e compreende a implementação das regras de introdução dos operadores.

```
proof(ctx(H,O),['id']) :- member(O,H).
proof(ctx(H,O),RS) :- deduce(ctx(H,O),RS).
proof(ctx(H,and(A,B)),['and-i'|R]) :- proof(ctx(H,A),RS),
                                      proof(ctx(H,B),RS1),
                                      append(RS,RS1,R).
proof(ctx(H,imp(A,B)),['intro-imp'|RS]) :- proof(ctx([A|H],B),RS).
proof(ctx(H,or(A,B)),['i-or'|RS]) :- proof(ctx(H,A),RS).
proof(ctx(H,or(A,B)),['i-or'|RS]) :- proof(ctx(H,B),RS).
```

Para implmentar este trabalho realize as seguintes tarefas:

1. Implmente as regras que faltam no sistema.
2. Existem algun exemplo de sequente que faz com que o programa entre em loop ?
3. Proponha uma melhoria que permita ao sistema provar sequentes como: $\{(A \rightarrow B) \rightarrow C, A, B\} \vdash C$