Elementos Lógicos da Programação I– Módulo II

Sistemas Dedutivos de Gentzen

1. Introdução.
2. Dedução Natural: os sistemas $\mathbf{N}_c$ e $\mathbf{Ns}_c$ e as versões intuicionistas $\mathbf{N}_i$ e $\mathbf{Ns}_i$.
3. Equivalência de Sistemas de Dedução da Lógica Proposicional.
4. Provas por propagação em retrocesso. Resolução.
5. Noção generalizada de sequente e sua validade.
6. O cálculo de sequentes LK+(CORTE).
7. Equivalência entre sistemas de dedução natural e o sistema **LK**.
8. O sistema LJ.

José Manuel E. Valença @ DI Universidade do Minho 1999-2000

Introdução

Os sistemas de dedução hilbertianos são de difícil utilização uma vez que a definição de algoritmos que automatizem (ou, pelo menos, sistematizem) a construção de deduções é muito complicada e os resultados desses eventuais algoritmos são pouco satisfatórios.

O problema essencial da dedução hilbertiana é o facto de não prever o uso de *assunções auxiliares* que possam ser introduzidas ou retiradas nos vários passos da dedução.

Exemplo: o modo "*natural*" de provar o teorema $P \supset P$ seria

(1) assume-se a validade de P;

(2) infere-se P de (1);

(3) fecha-se a assunção P e obtém-se $P \supset P$ (ex: o teorema da dedução).

Neste exemplo ilustra-se a funcionalidade básica necessária para lidar com assunções de uma forma que nos parece "*natural*": criar uma assunção, inferir a partir dela e, finalmente, fechar o uso da assunção.

Um sistema de dedução que se baseia nesta funcionalidade foi criado por Gerhard Gentzen nos anos 30 e chama-se **dedução natural**.

Aqui as assunções são vistas como um conjunto global de hipóteses: cada passo de dedução têm associada uma conclusão dentro desse conjunto global de assunções e transforma esse conjunto.

Em alternativa Gentzen criou uma segunda família de sistemas de dedução onde as assunções têm um carácter local: a conclusão de cada passo de dedução tem associado um conjunto próprio de assunções.

Esses objectos (conclusão associada a hipótese) chamaremos **sequente** e o sistema de dedução daí resultante chama-se **cálculo de sequentes**.

Dedução Natural

Como resultado de ser orientado à assunção e não à conclusão (como o sistema hilbertiano) a dedução natural é definida, essencialmente, por *regras de inferência* que sistematizam a funcionalidade base para lidar com assunções.

Os axiomas nos sistemas de Gentzen são só regras sem premissas.

Uma **regra de inferência** na dedução natural escreve-se

$$\frac{\begin{array}{cccc}[A_1] & [A_2] & & [A_n] \\ P_1 & P_2 & \cdots & P_n\end{array}}{C}\ \langle \mathsf{nome} \rangle$$

Além das premissas P_i e da conclusão C que têm um significado idêntico nas regras do sistema hilbertiano, ocorrem aqui **assunções** A_i. A proposição A_i é a *assunção a fechar na premissa* P_i pela aplicação desta regra.

Exemplo: Duas das regras paradigmáticas de dedução natural são

$$\frac{\begin{array}{c}[P] \\ Q\end{array}}{P \supset Q}\ \mathsf{I}_{\supset} \qquad\qquad \frac{\begin{array}{ccc}[P] & [Q] & \\ C & C & P \vee Q\end{array}}{C}\ \mathsf{E}_{\vee}$$

A regra $\mathsf{I}_{\supset}$ (lê-se *introdução* de $\supset$) fecha a hipótese P na premissa Q para produzir a conclusão $P \supset Q$.

A regra $\mathsf{E}_{\vee}$ (lê-se *eliminação* de $\vee$) fecha a assunção P na 1ª premissa e a assunção Q na 2ª premissa. A terceira premissa $P \vee Q$ não fecha qualquer assunção.

Toda a regra de dedução natural tem uma **fórmula principal** gerada por um dos conectivos da lógica. Em $\mathsf{I}_{\supset}$ a fórmula principal é $P \supset Q$ enquanto que $P \vee Q$ é a fórmula principal de $\mathsf{E}_{\vee}$.

Uma **regra de introdução** tem a fórmula principal como conclusão; uma **regra de eliminação** tem a fórmula principal como premissa.

8 DEFINIÇÃO *Uma dedução* $\Gamma \vdash P$ *, no sistema dedução natural, é uma árvore que satisfaz as seguintes condições:*

(1) Cada folha da árvore ou é uma proposição $A \in \Gamma$ *(e, neste caso, diz-se que é uma* ***assunção aberta****) ou então é uma uma proposição arbitrária* A *associada a uma* marca *e, neste caso, chama-se* ***assunção fechada****.*

A marca serve para distinguir uma assunção aberta de uma assunção fechada (mesmo que descritas pela mesma proposição) e serve também para distinguir duas assunções fechadas com a mesma proposição.

(2) Cada nodo interno C *é conclusão de uma regra de dedução*

$$\frac{\begin{array}{cccc}[A_1] & [A_2] & & [A_n] \\ P_1 & P_2 & \cdots & P_n\end{array}}{C}$$

tal que:

(a) Cada premissa P_i *é conclusão de uma dedução* $\mathcal{D}_i$

(b) As sub-árvores de C *são determinadas pelas premissas* P_i *e coincidem com a dedução* $\mathcal{D}_i$ *respectiva excepto em que todas as eventuais ocorrências aí de assunções abertas* A_i *são convertidas em assunções fechadas (e marcadas apropriadamente).*

Exemplo 1 uma árvore de dedução para $P \supset P$ pode ser construída do seguinte modo

1. Primeiro constrói-se uma árvore formada por uma só folha P; esta é uma árvore que tem P não só como conclusão mas também como assunção aberta.
2. Usa-se a regra $\mathsf{I}_{\supset}$ que introduz a conclusão $P \supset P$ e fecha a assunção P

$$\frac{\overline{P}}{P \supset P}\ \mathsf{I}_{\supset}$$

Para indicar o fecho da assunção P coloca-se um traço sobre a assunção.

A árvore resultante (muito simples, com apenas dois nodos) não tem assunções abertas e tem conclusão $P \supset P$; denota, assim, a relação $\{\} \vdash P \supset P$.

Exemplo 2 Uma dedução para o teorema $P \vee P \supset P$

$$\dfrac{\dfrac{\dfrac{}{P}\ (1) \qquad \dfrac{}{P}\ (1) \qquad \dfrac{}{P \vee P}\ (2)}{P}\ \mathsf{E}_{\vee}(1)}{P \vee P \supset P}\ \mathsf{I}_{\supset}(2)$$

Parte-se de três assunções abertas: P, de novo P e $P \vee P$.

Usa-se a regra $\mathsf{E}_{\vee}$ que introduz a conclusão P e fecha as duas assunções P. Nas assunções fechadas coloca-se a marca (1) que aparece também na regra.

Neste estado temos ainda aberta a assunção $P \vee P$ e temos P como conclusão. Usando a regra $\mathsf{I}_{\supset}$ fecha-se essa assunção (marcada com (2)) e introduz-se a conclusão $P \vee P \supset P$.

Obteve-se uma dedução sem assunções abertas e com a conclusão pretendida.

Regras da Dedução Natural

Regras de Introdução

$$\dfrac{P \quad Q}{P \wedge Q}\ \mathsf{I}_{\wedge} \qquad \dfrac{\begin{matrix}[P]\\ Q\end{matrix}}{P \supset Q}\ \mathsf{I}_{\supset} \qquad \dfrac{P}{P \vee Q}\ \mathsf{I}_{\vee 1} \qquad \dfrac{Q}{P \vee Q}\ \mathsf{I}_{\vee 2}$$

Regras de Eliminação

$$\dfrac{P \wedge Q}{P}\ \mathsf{E}_{\wedge 1} \qquad \dfrac{P \wedge Q}{Q}\ \mathsf{E}_{\wedge 2} \qquad \dfrac{P \quad P \supset Q}{Q}\ \mathsf{E}_{\supset} \qquad \dfrac{\begin{matrix}[P] & [Q]\\ C & C\end{matrix} \quad P \vee Q}{C}\ \mathsf{E}_{\vee}$$

Regras do Absurdo e Negação

$$\dfrac{\bot}{P}\ (\mathsf{ExF}) \qquad \dfrac{\begin{matrix}[\neg P]\\ P\end{matrix}}{P}\ (\mathsf{RAb}) \qquad \neg P \equiv P \supset \bot$$

Notas

1. O primeiro grupo apresenta regras de *introdução* para os três conectivos binários: $\wedge$, $\vee$ e $\supset$. Em cada uma de regras a fórmula principal (a que contém o conectivo que caracteriza a regra) é *introduzido na conclusão*.

   Note-se que a regra $\mathsf{I}_{\supset}$ determina, essencialmente um teorema de dedução: deduzir $P \supset Q$ é equivalente a deduzir Q sob hipótese P.

2. As regras de eliminação constituem o 2º grupo e nelas a fórmula principal ocorre numa premissa; isto significa que o objectivo é *eliminar o conectivo na conclusão.*

   Note-se que a regra de eliminação da implicação ($\mathsf{E}_{\supset}$) é a regra *Modus Ponens* que é a única regra do sistema hilbertiano.

3. O 3º conjunto de regras lida, de forma específica, com o conectivo absurdo $\perp$ e, através dele, com a negação: a negação $\neg P$ é vista como uma abreviatura para a implicação $P \supset \perp$.

   São duas as regras que caracterizam o absurdo e, simultaneamente, o tipo de lógica proposicional que o sistema descreve.

   (a) A primeira regra, representada por (ExF), é conhecida por *Ex Falsum Quod Libet* ou, simplesmente, *Ex Falsum.*

   Essencialmente diz-nos que de uma dedução que conclua o absurdo se pode concluir qualquer outra proposição: uma inferência de premissa absurda é sempre válida qualquer que seja a conclusão.

   (b) A segunda regra é o conhecido *Reductio Ad Absurdum* ou, **lei da redução ao absurdo**:

   Se a assunção de $\neg P$ permite deduzir uma conclusão absurda então pode-se inferir P.

   O *Sistema de Dedução Natural para a Lógica Proposicional Clássica*, abreviadamente representado por $\mathbf{N}_c$, é determinado pela globalidade das regras aqui apresentadas.

   O *Sistema de Dedução Natural para a Lógica Proposicional Intuicionista*, abreviadamente representado por $\mathbf{N}_i$ é determinado por todas as regras excepto a última (RAb).

Construção de Provas

Uma vez estabelecido a noção de dedução torna-se necessário sistematizar o processo de construção dessas deduções. Como ?

A abordagem mais directa será a que segue exactamente a definição de dedução natural e está representada no seguinte programa PROLOG.

```
natural([C],C,[C]).
natural(Ass,C,[C|Ds]) :- regra(C,Ps) , naturals(Acc,Ps,Ds).

naturals(Ass,[],[]).
naturals(Ass,[[P,A]| Ps],[D|Ds]) :-
      natural([A|Ass],P,D), naturals(Ass,Ps,Ds).
naturals(Ass,[P|Ps],[D|Ds])      :-
      natural(Ass,P,D), naturals(Ass,Ps,Ds).
```

O programa usa uma base de factos, estabelecida pelo predicado `regra`, que define as regras de inferência. Neste exemplo usamos apenas as regras de introdução e da eliminação da disjunção ($\vee$) e da implicação ($\supset$).

```
regra(P => Q, [[Q,P]]). regra(P v Q,[P]). regra(P v Q,[Q]).
regra(Q, [P , P => Q]). regra(C, [[C,P] , [C,Q] , P v Q]).
```

Em deduções onde só são usadas regras de introdução este programa não tem dificuldade em encontrar a solução correcta.

```
| ?- natural([], p => p, D).
D = [p=>p,[p]] ?
yes
| ?- natural([], p => p v q, D).
D = [p=>p v q,[p v q,[p]]] ?
yes
```

Porém, se a dedução exige uma regra de eliminação, este programa não consegue calculá-la.

O programa PROLOG tenta construir a dedução partindo da conclusão para as premissas e tenta aplicar a cada uma das premissas o mesmo processo que aplicou à conclusão.

Genericamente esta abordagem para construção de provas designa-se por **propagação em retrocesso**.

§ Na construção de provas por propagação em retrocesso é crucial que as regras satisfaçam a chamada **propriedade da sub-fórmula**: nas premissas as eventuais fórmulas principais são sempre sub-fórmulas de fórmulas da conclusão.

Quando esta propriedade se verifica, uma prova por propagação em retrocesso (progredindo da conclusão para as premissas) vai sempre simplificando fórmulas até estas serem atómicas; assim garante-se que a prova termina.

Na dedução natural as regras de introdução satisfazem a propriedade da sub-fórmula mas as de eliminação não verificam esta propriedade.

Basta ver a forma de uma das regras (por exemplo, $\mathsf{E}_\vee$) tem, nas premissas, fórmulas que não ocorrem na conclusão. A propagação em retrocesso, ao evoluir da conclusão para as premissas, irá "complicar" neste caso a fórmula principal.

Em alternativa é possível construir uma prova evoluindo das premissas para as conclusões; isto designa-se por **propagação em progresso**.

§ As provas construídas por propagação em progresso não são afectadas (directamente) pela verificação, pelas regras, da propriedade de sub-fórmula.

Aqui, porém, é necessário prever quais as assunções que podem vir a produzir a conclusão pretendida. A prova é muito mais difícil de sistematizar porque não existe a estrutura da conclusão para guiar a escolha da regra de inferência.

É muito mais simples construir uma "prova manual" (porque podemos usar heurísticas e experiência para guiar a escolha da regra de inferência e das assunções adequadas a cada passo) do que um algoritmo automático que tem dificuldade, normalmente, em usar tais heurísticas.

Exemplo 1: construção, por propagação em progresso, da dedução natural do teorema $((P \supset Q) \supset P) \supset P$.

Parte-se de três assunções: $\neg P$, P e $(P \supset Q) \supset P$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg P}\,(2) \quad \dfrac{}{P}\,(1)}{\bot}\,\mathsf{E}_{\supset}}{Q}\,(\mathsf{ExF})}{P \supset Q}\,\mathsf{I}_{\supset}(1) \qquad \dfrac{}{(P \supset Q) \supset P}\,(3)}{\dfrac{P}{P}\,(\mathsf{RAb})(2)}\,\mathsf{E}_{\supset}}{((P \supset Q) \supset P) \supset P}\,\mathsf{I}_{\supset}(3)$$

Esta forma de construção tem a grande desvantagem de não ter um "guia" que permita escolher, no início, quais as assunções que devem ser colocadas e decidir, em cada estado subsequente, qual a regra a aplicar de modo a se chegar à conclusão pretendida e, ao mesmo tempo, se conseguir fechar todas as assunções definidas de início.

De qualquer forma a árvore aqui construída poderia ser representada de forma alternativa. Usamos a indentação para representar os níveis da árvore e, junto a cada conclusão, indicamos o conjunto de assunções abertas nesse estado.

Pierce	$((P \supset Q) \supset P) \supset P$	
1. $\neg P$	$[(P \supset Q) \supset P, \neg P, P]$	assunção
2. P	$[(P \supset Q) \supset P, \neg P, P]$	assunção
1. $\bot$	$[(P \supset Q) \supset P, \neg P, P]$	$\mathsf{E}_{\supset}$
1. Q	$[(P \supset Q) \supset P, \neg P, P]$	ExF
1. $P \supset Q$	$[(P \supset Q) \supset P, \neg P]$	$\mathsf{I}_{\supset}$
2. $(P \supset Q) \supset P$	$[(P \supset Q) \supset P, \neg P]$	assunção
1. P	$[(P \supset Q) \supset P, \neg P]$	$\mathsf{E}_{\supset}$
1. P	$[(P \supset Q) \supset P]$	RAb
1. $((P \supset Q) \supset P) \supset P$	$[\,]$	$\mathsf{I}_{\supset}$

Normalmente não é necessário ter presente, em cada estado da construção da prova, todas as assunções que estão abertas: basta normalmente representar aquelas que são "relevantes" para o estado em questão e para todos os estados subsequentes.

Dentro deste princípio poderíamos representar o quadro anterior da seguinte forma

Pierce	$((P \supset Q) \supset P) \supset P$	
1. $\neg P$	$[\neg P]$	assunção
2. P	$[P]$	assunção
1. $\bot$	$[\neg P, P]$	$\mathsf{E}_{\supset}$
1. Q	$[\neg P, P]$	ExF
1. $P \supset Q$	$[\neg P]$	$\mathsf{I}_{\supset}$
2. $(P \supset Q) \supset P$	$[(P \supset Q) \supset P]$	assunção
1. P	$[(P \supset Q) \supset P, \neg P]$	$\mathsf{E}_{\supset}$
1. P	$[(P \supset Q) \supset P]$	RAb
1. $((P \supset Q) \supset P) \supset P$	$[\,]$	$\mathsf{I}_{\supset}$

Genericamente as diferentes proposições/conclusões "herdam" as assunções dos níveis superiores que não sejam fechadas pela aplicação da regra.

Este quadro representa essencialmente a mesma informação que o quadro anterior numa abordagem mais sistemática à representação de assunções. Abre, no entanto, perspectiva a duas decisões com grande importância:

(1) Se cada nodo na árvore tem associado o seu conjunto próprio de assunções faz sentido agregar conclusão+assunções numa <u>única entidade</u> que descreva não só uma frase que queiramos validar mas também as hipóteses usadas nessa validação.

(2) A agregação conclusão+assunções permite ver a fórmula principal de um passo de prova uma vez englobado nas assunções e outras vezes incluído na conclusão. Isto vai permitir que a estrutura sintáctica dessas agregações "guie" o processo de prova.

A agregação de um conjunto de assinções Δ com uma conclusão C cria uma entidade a que chama-mos **sequente** e que se escreve [a]

$$\Delta \Rightarrow C$$

Em termos de sequentes a dedução natural representada no quadro anterior será

Pierce	$((P \supset Q) \supset P) \supset P$	
1.	$\neg P \Rightarrow \neg P$	assunção
2.	$P \Rightarrow P$	assunção
1.	$\neg P, P \Rightarrow \bot$	$\mathsf{E}_\supset$
1.	$\neg P, P \Rightarrow Q$	ExF
1.	$\neg P \Rightarrow (P \supset Q)$	$\mathsf{I}_\supset$
2.	$((P \supset Q) \supset P) \Rightarrow ((P \supset Q) \supset P)$	assunção
1.	$((P \supset Q) \supset P), \neg P \Rightarrow P$	$\mathsf{E}_\supset$
1.	$((P \supset Q) \supset P) \Rightarrow P$	RAb
1.	$\Rightarrow ((P \supset Q) \supset P) \supset P$	$\mathsf{I}_\supset$

Este quadro descreve uma nova árvore em que, agora, os nodos deixam de ser proposições para passarem a ser sequentes:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\textsf{ass}}{\neg P \Rightarrow \neg P} \quad \dfrac{\textsf{ass}}{P \Rightarrow P}}{\neg P, P \Rightarrow \bot}\,(\mathsf{E}_\supset)}{\neg P, P \Rightarrow Q}\,(\mathsf{ExF})}{\neg P \Rightarrow (P \supset Q)}\,(\mathsf{I}_\supset) \qquad \dfrac{\textsf{ass}}{((P \supset Q) \supset P) \Rightarrow ((P \supset Q) \supset P)}}{((P \supset Q) \supset P), \neg P \Rightarrow P}\,(\mathsf{E}_\supset)}{\dfrac{((P \supset Q) \supset P) \Rightarrow P}{\Rightarrow ((P \supset Q) \supset P) \supset P}\,\mathsf{I}_\supset}\,(\mathsf{RAb})$$

[a]Existem várias formas de escrever este mesmo sequente; uma delas $\Delta \vdash C$ é usada no sistema Jape. No entanto, porque neste curso $\vdash$ já representa relações de dedução, escolhemos $\Rightarrow$ para os sequentes.

Apesar das aparências a dedução natural escrita em termos de sequentes é bastante mais simples que a dedução original porque não é preciso gerir informação global a toda a dedução (como nas operações de fecho de assunções). Toda a manipulação é agora feita localmente quer no lado das assunções quer do lado da conclusão.

Torna-se, no entanto, necessário definir um novo conceito de *dedução*, uma nova sintaxe de *regra de inferência* e um novo *conjunto de regras* de inferência de acordo com essa nova sintaxe.

9 DEFINIÇÃO *Uma dedução natural em sequentes, para a relação* $\Gamma \vdash C$, *é uma árvore cujos nodos são sequentes, cuja conclusão é o sequente* $\Gamma \Rightarrow C$ *e que satisfaz uma das seguintes condições:*

(1) A árvore é uma folha $\Gamma \Rightarrow C$ *com* $C \in \Gamma$*; neste caso a dedução diz-se justificada por* **assunção**.

(2) Existe uma regra de inferência

$$\frac{\Delta_1 \Rightarrow P_1 \quad \Delta_2 \Rightarrow P_2 \quad \cdots \quad \Delta_n \Rightarrow P_n}{\Gamma \Rightarrow C}$$

e existe dedução $\mathcal{D}_i$ *para cada uma das premissas* $\Delta_i \Rightarrow P_i$*; a dedução tem* $\Gamma \Rightarrow C$ *como raiz e tem as várias deduções* $\mathcal{D}_i$ *como sub-árvores.*

Note-se, em primeiro lugar, a simplicidade desta definição quando comparada com a definição 8.

Note-se também que, porque não existe gestão global de assunções, a definição determina uma estrutura simplesmente definida recursivamente; é natural, por isso, que os algoritmos que manipulam esta estrutura tenham uma forma recursiva simples.

O **sistema de regras de inferência** para a **dedução natural em sequentes** da Lógica Proposicional é agora definido da seguinte forma: [a]

Regras de Introdução

$$\frac{\Gamma \Rightarrow P \qquad \Delta \Rightarrow Q}{\Gamma, \Delta \Rightarrow P \wedge Q}\ (\mathsf{I}_\wedge) \qquad \frac{\Gamma, P \Rightarrow Q}{\Gamma \Rightarrow (P \supset Q)}\ (\mathsf{I}_\supset)$$

$$\frac{\Gamma \Rightarrow P}{\Gamma \Rightarrow P \vee Q}\ (\mathsf{I}_\vee 1) \qquad \frac{\Gamma \Rightarrow Q}{\Gamma \Rightarrow P \vee Q}\ (\mathsf{I}_\vee 2)$$

Regras de Eliminação

$$\frac{\Gamma \Rightarrow P \wedge Q}{\Gamma \Rightarrow P}\ (\mathsf{E}_\wedge 1) \qquad \frac{\Gamma \Rightarrow P \wedge Q}{\Gamma \Rightarrow Q}\ (\mathsf{E}_\wedge 2)$$

$$\frac{\Gamma \Rightarrow P \quad \Delta \Rightarrow P \supset Q}{\Gamma, \Delta \Rightarrow Q}\ (\mathsf{E}_\supset) \qquad \frac{\Gamma, P \Rightarrow C \quad \Delta, Q \Rightarrow C \quad \Upsilon \Rightarrow P \vee Q}{\Gamma, \Delta, \Upsilon \Rightarrow C}\ (\mathsf{E}_\vee)$$

Regras do Absurdo e Negação

$$\frac{\Gamma \Rightarrow \bot}{\Gamma \Rightarrow P}\ (\mathsf{ExF}) \qquad \frac{\Gamma, \neg P \Rightarrow P}{\Gamma \Rightarrow P}\ (\mathsf{RAb}) \qquad \neg P \equiv (P \supset \bot)$$

Representamos por $\mathbf{Ns}_c$ o sistema de dedução natural orientado ao sequente para a Lógica Proposicional Clássica determinado pela definição 9 e por este conjunto de 8 regras de inferência; representamos por $\mathbf{Ns}_i$ a sua versão intuicionista onde se exclui a regra (RAb).

[a] A vírgula é usada tanto para representar a união de dois conjuntos de assunções (ex: Γ, Δ) como a junção de uma assunção a um conjunto (ex: Γ, P).

Equivalência de Sistemas de Dedução da Lógica Proposicional

Neste curso foram já apresentados três sistemas de dedução para a Lógica Proposicional Clássica: o sistema **H** de dedução hilbertiano (definição 7), o sistema $\mathbf{N}_c$ de dedução natural orientado às fórmulas (definição 8) e o sistema $\mathbf{Ns}_c$ de dedução natural baseado em sequentes (definição 9).

Torna-se necessário comparar as relações de dedução determinadas por estes sistemas. Vamos verificar que eles são, essencialmente, equivalentes: isto é, provam exactamente os mesmos teoremas.

6 Proposição *Os sistemas de dedução* $\mathbf{N}_c$ *e* $\mathbf{Ns}_c$ *são equivalentes: uma proposição é teorema num dos sistemas se e só se for teorema no outro. Analogamente são equivalentes os sistemas* $\mathbf{N}_i$ *e* $\mathbf{Ns}_i$*.*

Prova

Os dois sistemas diferem apenas no tratamento das assunções. No sistema $\mathbf{Ns}_c$ a conclusão acumula sempre as assunções abertas das premissas que não sejam fechadas explicitamente pela regra; por isso, em qualquer estado da prova, $\Gamma \Rightarrow P$, o lado esquerdo Γ acumulam todas as assunções abertas na árvore acima. Isto é Γ coincide com o conjunto global de assunções abertas na prova $\mathbf{N}_c$ correspondente (com conclusão P).

Feita esta observação, se tomarmos qualquer dedução em $\mathbf{Ns}_c$ de conclusão $\Gamma \Rightarrow P$, obtemos uma dedução em **N** de conclusão P usando as regras de inferência correspondentes e Γ como conjunto global de assunções abertas nesse estado.

Inversamente se tomarmos qualquer dedução em $\mathbf{N}_c$ podemos, em cada estado P, reduzir o conjunto de assunções abertas às assunções que são introduzidas na sub-árvore de raiz P (tal como fizemos na construção do último quadro).

Desta forma construímos o conjunto Γ das assunções "locais" a P e, daí, o sequente $\Gamma \Rightarrow P$. Usando as regras correspondentes às usadas em $\mathbf{N}_c$ constrói-se uma dedução em $\mathbf{Ns}_c$ que tem esse sequente como conclusão.

Os mesmos argumentos são aplicáveis aos sistemas intuicionistas $\mathbf{N}_i$ e $\mathbf{Ns}_i$.

Formalmente as propriedades essenciais dos dois sistemas de dedução natural $\mathbf{N}_c$ e $\mathbf{Ns}_c$ são expressos nos seguintes resultados.

3 Teorema *Os sistemas de dedução* $\mathbf{H}$, $\mathbf{N}_c$ *e* $\mathbf{Ns}_c$ *são equivalentes.*

Prova
A prova da equivalência entre $\mathbf{H}$ e $\mathbf{N}_c$ é apresentada no texto *Basic Proof Theory* (teorema 2.4.3) referido na apresentação deste curso.

Essencialmente a prova analisa as várias possibilidades de chegar a uma conclusão P, num dos sistemas, e atendendo à estrutura sintáctica de P constrói uma prova no outro sistemas. A equivalência entre $\mathbf{H}$ e $\mathbf{Ns}_c$ resulta desta primeira equivalência e da proposição 6.

3 Corolário Os sistemas de dedução $\mathbf{N}_c$ e $\mathbf{Ns}_c$ para a Lógica Proposicional Clássica são correctos e completos.

Prova
O sistema $\mathbf{H}$ é correcto e completo (teorema 2); como é equivalente a $\mathbf{N}_c$ e a $\mathbf{Ns}_c$ ambos estes sistemas de dedução têm de ser igualmente correctos e completos.

4 Corolário As relações de dedução definidas pelos sistemas de dedução $\mathbf{N}_c$ e $\mathbf{Ns}_c$ para a Lógica Proposicional Clássica satisfazem as condições de **inclusão**, **monotonia** e **corte**.

Prova
A relação de dedução determinada por $\mathbf{N}_c$ (ou $\mathbf{Ns}_c$) coincide, pelo teorema 3, com a determinada pelo sistema $\mathbf{H}$; como esta última satisfaz as condições de inclusão, monotonia e corte, a primeira também satisfaz as mesmas condições.

Propagação em retrocesso no sistema $\mathbf{Ns}_c$

Após estabelecer a equivalência, correcção e completude dos sistemas de dedução natural é necessário voltar a analisar a questão da *sistematização da prova*: como definir, se possível, uma *algoritmo* que, partindo do conjunto de hipótese Γ e da conclusão P, construa uma dedução para a relação $\Gamma \vdash P$ num dos sistemas de dedução natural .

Dado que a sistematização de uma prova por propagação em progresso é difícil de realizar, procuraremos apresentar construções de prova baseadas, essencialmente, na **propagação em retrocesso**.

As provas constroem deduções "*da conclusão para as premissas*": as árvores que as representam são construídas "*da raiz para as folhas*".

A construção, por propagação em retrocesso, da dedução que apresentámos para a tautologia $((P \supset Q) \supset P) \supset P$ pode ser inicializada no seguinte quadro

Pierce	$((P \supset Q) \supset P) \supset P$	
1.	$\Rightarrow ((P \supset Q) \supset P) \supset P$	$\mathsf{I}_\supset$
1.	$(P \supset Q) \supset P \Rightarrow P$	RAb
1.	$\neg P\,,\, (P \supset Q) \supset P \Rightarrow P$	$\cdots$

que corresponde à árvore

$$\dfrac{\dfrac{\neg P\,,\, (P \supset Q) \supset P \Rightarrow P}{(P \supset Q) \supset P \Rightarrow P}\ (\mathsf{RAb})}{\Rightarrow ((P \supset Q) \supset P) \supset P}\ (\mathsf{I}_\supset) \qquad (1)$$

Esta árvore **não é** uma dedução em $\mathbf{Ns}_c$ porque a sua única folha não verifica nenhuma das condições previstas na definição 9; a árvore representa uma *dedução parcial* (ou *incompleta*) e designa um **estado de prova**.

A propagação em retrocesso ***transforma*** *estados de prova noutros estados de prova até atingir a dedução final.*

Partindo de um estado de prova inicial definido pelo teorema que se pretende provar, este estado de prova foi alcançado por uma sequência de transformações:

$$(1) \quad \Rightarrow ((P \supset Q) \supset P) \supset P \qquad (2) \quad \frac{(P \supset Q) \supset P \Rightarrow P}{\Rightarrow ((P \supset Q) \supset P) \supset P}\ (\mathsf{I}_{\supset})$$

$$(3) \quad \frac{\dfrac{\neg P\,,\, (P \supset Q) \supset P \Rightarrow P}{(P \supset Q) \supset P \Rightarrow P}\ (\mathsf{RAb})}{\Rightarrow ((P \supset Q) \supset P) \supset P}\ (\mathsf{I}_{\supset}) \qquad (4) \quad \cdots$$

A transformação genérica aqui usada é condicionada por regras de inferência mas tem uma forma única já discutida na definição 9 (condição 2.); chama-se **resolução** e para a definir é conveniente apresentar alguns conceitos prévios.

10 DEFINIÇÃO *Um* ***estado de prova*** *para a relação* $\Gamma \vdash C$ *é uma árvore cujos nodos são sequentes e assume uma das formas:*

(1) *É uma folha* $\Gamma \Rightarrow C$.

(2) *Existe uma regra de inferência* $\dfrac{\Delta_1 \Rightarrow P_1 \quad \cdots \quad \Delta_n \Rightarrow P_n}{\Gamma \Rightarrow C}$ *e estados de prova* $\mathcal{D}_i$ *para cada uma das relações* $\Delta_i \vdash P_i$. *O estado de prova é formado pela raiz* $\Gamma \Rightarrow C$ *e pelas sub-árvores* $\mathcal{D}_i$.

Comparando esta definição com a de dedução (definição 9) constata-se que a diferença essencial está na exigência das folhas serem *fechadas por assunção* (i.e., $C \in \Gamma$) que é imposta na definição de dedução mas não é imposta na definição de estado de prova.

Deste modo,

Um estado de prova torna-se uma dedução quando todas as suas folhas são fechadas por assunção
Para construir uma dedução para $\Gamma \vdash C$ *parte-se do estado de prova* $\Gamma \Rightarrow C$ *e, por transformações apropriadas, vão-se obtendo novos estados de prova até atingir um que tenha todas as folhas fechadas por assunção.*

Resolução de estados de prova

Suponhamos que temos dois estados de prova $\mathcal{R}$ e $\mathcal{S}$ em que a conclusão de $\mathcal{R}$ coincide com uma das folhas de $\mathcal{S}$ (por exemplo, a primeira).

$$(\mathcal{R})\ \frac{R_1 \quad \ldots \quad R_m}{S_1} \qquad \frac{S_1 \quad \ldots \quad S_n}{S_0}\ (\mathcal{S})$$

Nestas circunstâncias diz-se que os dois estados **são resolúveis** e a sua **resolução** (representada por $\mathcal{R} \otimes \mathcal{S}$) é o estado de prova que se obtém de $\mathcal{S}$ substituindo a folha S_1 por $\mathcal{R}$.

$$\frac{\dfrac{R_1 \quad \ldots \quad R_m}{S_1} \qquad S_n}{S_0} \qquad (\mathcal{R} \otimes \mathcal{S})$$

Como forma essencial de transformação do estado de prova, as provas por retrocesso usam uma forma particular de resolução: consideram que $\mathcal{R}$ é sempre uma regra de inferência

$$(\mathcal{R})\ \frac{R_1 \quad \cdots \quad R_m}{S_1}$$

Isto é possível porque, de acordo com a definição, uma regra de inferência é um caso particular de dedução incompleta.

Nestas transformações $\mathcal{S}$ denota o estado que está a evoluir para a dedução pretendida. A resolução $\mathcal{R} \otimes \mathcal{S}$ será

$$\frac{\dfrac{R_1 \quad \cdots \quad R_m}{S_1} \qquad S_n}{S_0} \qquad (\mathcal{R} \otimes \mathcal{S})$$

Exemplo 1

Considere-se a construção de uma dedução para o teorema $P \vee P \supset P$.

Parte-se do estado de prova definido pelo sequente determinado pelo teorema

$$\Rightarrow P \vee P \supset P \qquad (*)$$

O estado é agora transformado por resolução com a regra $\mathsf{I}_{\supset}$ que, neste caso, toma a forma particular

$$\frac{P \vee P \Rightarrow P}{\Rightarrow P \vee P \supset P} \qquad (**)$$

Como o estado de prova é, por enquanto, uma simples folha a resolução coincide com esta mesma regra.

O estado seguinte obtém-se calculando a resolução do estado actual com a regra $\mathsf{E}_{\vee}$ na forma particular

$$\frac{P \vee P, P \Rightarrow P \qquad P \vee P, P \Rightarrow P \qquad P \vee P \Rightarrow P \vee P}{P \vee P \Rightarrow P} \qquad (\dagger)$$

O estado de prova resultante da resolução de $(\dagger)$ com $(**)$ será

$$\frac{\dfrac{P \vee P, P \Rightarrow P \qquad P \vee P, P \Rightarrow P \qquad P \vee P \Rightarrow P \vee P}{P \vee P \Rightarrow P}}{\Rightarrow P \vee P \supset P} \qquad (\ddagger)$$

Todas as folhas deste estado de prova são fechadas por assunção. Por isso o estado é a dedução pretendida.

Este mesmo processo de construção de uma dedução pode ser sumariado no seguinte quadro.

$\Rightarrow P \vee P \supset P$	
1. $\Rightarrow P \vee P \supset P$	$\mathsf{I}_{\supset}$
1. $P \vee P \Rightarrow P$	$\mathsf{E}_{\vee}$
1. $P \vee P, P \Rightarrow P$	assunção
2. $P \vee P, P \Rightarrow P$	assunção
3. $P \vee P \Rightarrow P \vee P$	assunção

Exemplo 2

Considere-se como estado de prova $\mathcal{S}$ a árvore em (1) (acetato 15) e como regra de inferência uma instância da regra $\mathsf{E}_{\supset}$.

A resolução com uma regra de eliminação introduz sempre uma proposição "*incógnita*" que podemos instanciar com uma proposição concreta da forma que for mais conveniente ao prosseguimento da prova.

Neste caso a regra, onde $?A$ denota essa proposição *incógnita*, é

$$(\mathcal{R}) \qquad \frac{\Delta \Rightarrow ?A \supset P \qquad \Delta \Rightarrow ?A}{\Delta \Rightarrow P} \qquad \text{com} \quad \Delta \equiv \{\neg P\,,\,(P \supset Q) \supset P\}$$

A regra não satisfaz a condição de sub-fórmula devido, precisamente, à ambiguidade em relação a $?A$. Note-se porém que, qualquer que seja $?A$, $\mathcal{R}$ e $\mathcal{S}$ são resolúveis.

A evolução do estado de prova pode ser representada no seguinte quadro.

Pierce	$((P \supset Q) \supset P) \supset P$	
1.	$\Rightarrow ((P \supset Q) \supset P) \supset P$	$\mathsf{I}_{\supset}$
1.	$(P \supset Q) \supset P \Rightarrow P$	RAb
1.	$\neg P\,,\,(P \supset Q) \supset P \Rightarrow P$	$\mathsf{E}_{\supset}$
1.	$\neg P\,,\,(P \supset Q) \supset P \Rightarrow ?A \supset P$	$\cdots$
2.	$\neg P\,,\,(P \supset Q) \supset P \Rightarrow ?A$	$\cdots$

Com a intensão de fechar sequentes por assunção, são possíveis diferentes escolhas para $?A$: qualquer das escolhas $?A \equiv \neg P$, ou $?A \equiv (P \supset Q) \supset P$ ou ainda $?A \equiv (P \supset Q)$, fecha uma das folhas "abertas".

A "boa escolha" deve facilitar a construção de uma prova para a folha restante. Por isso vamos adoptar pela última possibilidade: $?A \equiv (P \supset Q)$.

O quadro de prova é então ... (*continua*)

Pierce	$((P \supset Q) \supset P) \supset P$	
1.	$\Rightarrow ((P \supset Q) \supset P) \supset P$	$\mathsf{I}_\supset$
1.	$(P \supset Q) \supset P \Rightarrow P$	RAb
1.	$\neg P, (P \supset Q) \supset P \Rightarrow P$	$\mathsf{E}_\supset$
1.	$\neg P, (P \supset Q) \supset P \Rightarrow (P \supset Q) \supset P$	assunção
2.	$\neg P, (P \supset Q) \supset P \Rightarrow P \supset Q$	$\mathsf{I}_\supset$
1.	$P, \neg P, (P \supset Q) \supset P \Rightarrow Q$	ExF
1.	$P, \neg P, (P \supset Q) \supset P \Rightarrow \bot$	$\cdots$

Mais uma vez surge a necessidade de usar uma regra de eliminação; por coincidência é a mesma regra $\mathsf{E}_\supset$ já usada antes. O processo repete-se com a introdução de uma nova *proposição incógnita* agora representada por $?B$

Pierce	$((P \supset Q) \supset P) \supset P$	
	$\vdots$	$\vdots$
1.	$P, \neg P, (P \supset Q) \supset P \Rightarrow \bot$	$\mathsf{E}_\supset$
1.	$P, \neg P, (P \supset Q) \supset P \Rightarrow ?B \supset \bot$	$\cdots$
1.	$P, \neg P, (P \supset Q) \supset P \Rightarrow ?B$	$\cdots$

A escolha de $?B \equiv P$ e a identidade $\neg P \equiv P \supset \bot$ permite fechar por assunção as duas restantes folhas.

Pierce	$((P \supset Q) \supset P) \supset P$	
	$\vdots$	$\vdots$
1.	$P, \neg P, (P \supset Q) \supset P \Rightarrow \bot$	$\mathsf{E}_\supset$
1.	$P, \neg P, (P \supset Q) \supset P \Rightarrow \neg P$	assunção
1.	$P, \neg P, (P \supset Q) \supset P \Rightarrow P$	assunção

O seguinte programa PROLOG dá uma primeira aproximação à construção de provas por propagação em retrocessono sistema $\mathbf{Ns}_c$.

Os sequentes são representados por um par formado por uma lista de *assunções* e por uma *conclusão* separados pelo símbolo ==>. As regras são representadas por factos com três argumentos: o *nome*, a *conclusão* e uma lista de *premissas*.[a]

```
natural(G ==> C ,[G ==> C]) :- member(C,G).
natural(S,[S | SubArv])     :-
  print('regra para > '), print(S), nl, read(N), regra(N,S,Prems),
  naturals(Prems, SubArv).
naturals([],[]).
naturals([S|Ss],[A|As])   :- natural(S,A) , naturals(Ss,As).
%
regra(ii,  G ==> P => Q , [[P|G] ==> Q]).
regra(ei,  G ==> Q , [G ==> P => Q ,  G ==> P]).
regra(eo,  G ==> C , [[P|G] ==> C , [Q|G] ==> C , G ==> P v Q]).
regra(exf, G ==> Q ,  [G ==> P , G ==> -(P)]).
regra(rab, G ==> P ,  [[-(P)|G] ==> P]).
```

O programa interroga o utilizador sobre a regra que deve aplicar em cada passo: o utilizador "guia" o programa. Por exemplo, para provar $P \vee P \supset P$ o programa é guiado da seguinte forma:

```
| ?- natural([] ==> p v p => p, D).
regra para > []==>p v p=>p
|: ii.
regra para > [p v p]==>p
|: eo.
D = [[]==>p v p=>p,
       [[p v p]==>p,
           [[p,p v p]==>p],[[p,p v p]==>p],[[p v p]==>p v p]]] ?
yes
```

[a] Por simplicidade apresentamos aqui apenas algumas das regras.

Noção generalizada de sequente

Os sistemas de dedução natural orientados aos sequentes (por exemplo, $\mathbf{Ns}_c$) favorecem a construção de provas por propagação em retrocesso dada a simplicidade com que são definidos estados de prova e deduções completas.

No entanto, porque as regras de eliminação não satisfazem a propriedade da sub-fórmula, não existe garantia que as provas possam ser automatizadas: o uso dessas regras num estado de prova pode criar premissas mais complexas que a própria conclusão desse estado.

Como consequência seria ideal conseguir definir um sistema que definisse deduções simples (como $\mathbf{Ns}_c$) e, simultaneamente, só usasse regras de introdução.

Para construir um tal sistema pode-se usar o grau de liberdade, ainda não explorado, que consiste em introduzir as fórmulas principais tanto no lado esquerdo (assunções) como do lado direito (conclusões) do sequente.

Tomemos, como exemplo, a regra $\mathsf{E}_{\supset}$ no sistema $\mathbf{Ns}_c$

$$\frac{\Gamma, P \Rightarrow C \quad \Delta, Q \Rightarrow C \quad \Gamma, \Delta \Rightarrow P \vee Q}{\Gamma, \Delta \Rightarrow C}$$

Com a ajuda da regra *Modus Ponens* prova-se facilmente (como brevemente veremos) que esta regra é equivalente a uma outra

$$\frac{\Gamma, P \Rightarrow C \qquad \Delta, Q \Rightarrow C}{\Gamma, \Delta, P \vee Q \Rightarrow C}$$

Nesta nova regra constata-se que

(i) $(P \vee Q)$ é "transferida" de uma premissa para a conclusão da regra; a fórmula principal passa a ser parte da conclusão e, consequentemente, a regra transforma-se numa *regra de introdução*.

(ii) toda a manipulação de fórmulas passa a ser feita no lado das assunções (o lado direito dos sequentes não muda); nomeadamente $(P \vee Q)$ passa do lado direito da premissa para o lado esquerdo da conclusão.

Um sistema de dedução orientados aos sequentes que apenas usam regras de introdução designa-se, genericamente, por **cálculo de sequentes**.

Num cálculo de sequentes a noção de dedução e a noção de estado de prova são precisamente as mesmas que forma apresentadas, respectivamente, na definição 9 e na definição 10.

Porém a noção de sequente vai ser generalizada e o conjunto de regras de inferência será alterado para ser formado exclusivamente por regras que satisfazem a propriedade da sub-fórmula.

Ao contrário das regras de dedução natural, onde apenas as conclusões dos sequentes contém fórmulas principais de regras, nos cálculos de sequentes tanto as conclusões como as assunções dos sequentes contém fórmulas principais.

Para ser possível construir cálculos de sequentes que satisfaçam estes objectivos é necessário dar uma nova definição de sequente.

11 DEFINIÇÃO *Um **sequente** é um par de teorias* Γ *e* Δ *(eventualmente vazias) escrito* $\Gamma \Rightarrow \Delta$ *. Se* Δ *contém, quanto muito, uma fórmula o sequente diz-se **intuicionista**.*

Exemplo
As seguintes construções são sequentes

$$A, A \supset B \Rightarrow B \qquad A, \neg A \Rightarrow \qquad \Rightarrow A, \neg A$$

O primeiro é um sequente do tipo usado na dedução natural: um conjunto de fórmulas do lado esquerdo e uma única fórmula do lado direito.

Os dois restantes sequentes têm um dos lados vazio e nenhum deles se adequa à noção de sequente usada na dedução natural: num caso o lado direito não contém qualquer fórmula e no outro contém 2 fórmulas.

Os dois primeiros são sequentes intuicionistas mas o último não é.

É possível definir nos sequentes uma noção de **validade clássica** baseada na validade das fórmulas que o constituem. Porém, no estudo dos sequentes, é normalmente definido o conceito "*pela negativa*": isto é, procura-se determinar as circunstâncias em que o sequente não é válido. Neste caso o sequente diz-se *refutado.*

12 DEFINIÇÃO *Um modelo* $\mathcal{M}$ ***refuta o sequente*** $\Gamma \Rightarrow \Delta$ *se valida a teoria* $\Gamma\,, \neg\Delta$ *; isto é*

$$\mathcal{M} \models \Gamma\,, \neg\Delta$$

O sequente é ***refutado*** *se existe pelo menos um modelo que o refuta.*

Nota: dada uma teoria Δ representa-se por $\neg\Delta$ a teoria que e obtém substituindo toda a proposição $P \in \Delta$ por $\neg P$.

Por exemplo, $\neg\{A, \neg A\}$ é a teoria $\{\neg A, \neg\neg A\}$.

O modelo $\mathcal{M}$ valida a teoria $\Gamma, \neg\Delta$ se valida todas as suas proposições; isto significa que todas as fórmulas $P \in \Gamma$ devem ser válidas e todas as fórmulas $Q \in \Delta$ devem ser **não** válidas. Logo

$\Gamma \Rightarrow \Delta$ é refutado se existe um modelo que valide todas as fórmulas no lado esquerdo Γ e invalide todas as fórmulas do lado direito Δ.

Exemplo
Nenhum dos sequentes seguintes pode ser refutado

$$P \supset Q, P \Rightarrow Q \qquad \neg P, P \Rightarrow \qquad \Rightarrow \neg P, P \qquad P \Rightarrow P$$

No primeiro caso porque isso implicaria encontrar um modelo onde $P \supset Q$ e P fossem válidos e Q não válido.

No sequente $\neg P, P \Rightarrow$, porque isso implicaria encontrar um modelo onde $\neg P$ fosse válido e, ao mesmo tempo, P fosse também válido.

No sequente $\Rightarrow \neg P, P$, porque tal equivaleria a encontrar um modelo onde $\neg P$ e P fossem, simultaneamente, não válidas.

Finalmente $P \Rightarrow P$ não pode ser refutado porque isso equivaleria encontrar um modelo onde, simultaneamente, P fosse válido e não válido.

O predicado PROLOG `refuta` implementa a definição de refutação; tem por parâmetros o universo `U` dos símbolos proposicionais, um modelo `M` neste universo e o sequente `G ==> D`.

O predicado `neg` constrói a negação de uma teoria. O predicado `valt` é implementados separadamente e verifica a validade da teoria `T` no modelo `M`.

```
refuta(U,M, G ==> D) :-
  neg(D,Dn), append(G,Dn,T), sublist(M,U), valt(M,T).

neg([],[]).
neg([P|L],[-(P)|Ln]) :- neg(L,Ln).
```

Este programa determina se um sequente é refutado e, se for, determina um modelo que refuta o sequente. Como exemplos

```
| ?- refuta([p,q], M, [] ==> [((p => q) => p) => p]).
no
| ?- refuta([p,q], M, [p => q] ==> [q]).
M = [] ?
yes
| ?- refuta([p,q], M, [p => q] ==> [-(q)]).
M = [q,p] ?
yes
```

O primeiro exemplo mostra que $\Rightarrow ((P \supset Q) \supset P) \supset P$ não pode ser refutado e, por isso, serve de prova do lema de Pierce.

O segundo exemplo mostra que, num modelo onde P e Q não sejam válidos, o sequente $(P \supset Q) \Rightarrow Q$ é refutado.

Finalmente o terceiro exemplo mostra que um modelo que valide P e Q refuta o sequente $(P \supset Q) \Rightarrow \neg Q$.

Sistema de Dedução de Gentzen LK

A finalidade e um **cálculo de sequentes** é a obtenção de um sistema de dedução onde todas as regras de inferência sejam do tipo introdução.

Apenas assim é possível garantir que o sistema de dedução tenha a propriedade da sub-fórmula e por isso, numa prova por propagação em retrocesso, as fórmulas tornam-se progressivamente mais simples consoante a prova é construída.

As noções de dedução e de estado de prova são idênticas às definidas nas definições 9 e 10 mas adaptadas à versão generalizada de sequente.

13 DEFINIÇÃO *Um **estado de prova** para o sequente $\Gamma \Rightarrow \Delta$, no cálculo de sequentes, é uma árvore cujos nodos são sequentes e satisfaz uma das seguintes condições:*

(1) A árvore é a folha $\Gamma \Rightarrow \Delta$.

(2) Existe uma regra de inferência $\dfrac{\Gamma_1 \Rightarrow \Delta_1 \quad \cdots \quad \Gamma_n \Rightarrow \Delta_n}{\Gamma \Rightarrow \Delta}$ *e existe um estado de prova $\mathcal{D}_i$ para cada uma das premissas $\Gamma_i \Rightarrow \Delta_i$; a árvore tem $\Gamma \Rightarrow \Delta$ como raiz e os vários $\mathcal{D}_i$ como sub-árvores.*

*Um estado de prova é uma **dedução** para a relação $\Gamma \vdash C$ quando tem $\Gamma \Rightarrow C$ como conclusão e todas as folhas $\Upsilon \Rightarrow \Delta$ são fechadas por assunção; isto é, Υ e Δ contêm, pelo menos, uma proposição P comum.*

Os possíveis cálculos de sequentes vão ser determinados por conjuntos apropriados de regras de inferência.

Numa primeira abordagem vamos definir um conjunto de regras de inferência onde todas as regras são regras de introdução excepto uma.

Sistema de dedução LK+(corte)

Regra do Corte

$$\frac{\Gamma, C \Rightarrow \Delta \qquad \Gamma' \Rightarrow C, \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'} \text{ (corte)}$$

Regras "Direitas" (conclusões)

$$\frac{\Gamma \Rightarrow P, \Delta \qquad \Gamma \Rightarrow Q, \Delta}{\Gamma \Rightarrow P \wedge Q, \Delta} \ (\mathsf{D}_\wedge) \qquad \frac{\Gamma \Rightarrow P, Q, \Delta}{\Gamma \Rightarrow P \vee Q, \Delta} \ (\mathsf{D}_\vee)$$

$$\frac{\Gamma, P \Rightarrow Q, \Delta}{\Gamma \Rightarrow P \supset Q, \Delta} \ (\mathsf{D}_\supset) \qquad \frac{\Gamma, P \Rightarrow \Delta}{\Gamma \Rightarrow \neg P, \Delta} \ (\mathsf{D}_\neg)$$

Regras "Esquerdas" (assunções)

$$\frac{\Gamma, P, Q \Rightarrow \Delta}{\Gamma, P \wedge Q \Rightarrow \Delta} \ (\mathsf{E}_\wedge) \qquad \frac{\Gamma, P \Rightarrow \Delta \qquad \Gamma, Q \Rightarrow \Delta}{\Gamma, P \vee Q \Rightarrow \Delta} \ (\mathsf{E}_\vee)$$

$$\frac{\Gamma, Q \Rightarrow \Delta \qquad \Gamma \Rightarrow P, \Delta}{\Gamma, P \supset Q \Rightarrow \Delta} \ (\mathsf{E}_\supset) \qquad \frac{\Gamma \Rightarrow P, \Delta}{\Gamma, \neg P \Rightarrow \Delta} \ (\mathsf{E}_\neg)$$

Comentários

1. A primeira regra, conhecida por (corte), explicita a propriedade do corte em termos de sequentes.

   Vemos que nas premissas ocorre uma proposição C (dita *fórmula do corte*) que não ocorre na conclusão. Por isso esta regra não satisfaz a propriedade da sub-fórmula. De facto pretende-se que, em todo este sistema de dedução, ela seja a única que eventualmente não satisfaz essa propriedade.

2. Todas as restantes regras são regras de introdução (a fórmula principal ocorre sempre na conclusão) e dividem-se em dois grupos: as *regras direitas* são aquelas onde a fórmula principal ocorre no lado direito do sequente e as *regras esquerdas* são, naturalmente, aquelas onde essa fórmula ocorre no lado esquerdo.

3. As *regras direitas* manipulam conclusões de sequentes; as fórmulas principais nessas conclusões dão decompostas em sub-fórmulas que são passadas para qualquer dos lados das premissas. Vêm substituir as regras de introdução da dedução natural.

   As *regras esquerdas* vêm substituir as regras de eliminação da dedução natural. Lidam com as assunções das conclusões e transferem as sub-fórmulas da fórmula principal para qualquer dos lados das premissas.

4. Pela primeira vez introduzimos regras explícitas para a negação sem recorrer à intervenção do absurdo $\perp$. Neste sistema de dedução a negação é uma conectiva primitiva e o absurdo é visto como uma abreviatura

$$\perp \equiv P \wedge \neg P$$

Exemplo Prova de $((P \supset Q) \supset P) \supset P$ em LK+(CORTE).

Pierce	$((P \supset Q) \supset P) \supset P$

1. $\Rightarrow ((P \supset Q) \supset P) \supset P$	$\mathsf{D}_\supset$
1. $(P \supset Q) \supset P \Rightarrow P$	$\mathsf{E}_\supset$
1. $P \Rightarrow P$	assunção
2. $\Rightarrow P, P \supset Q$	$\mathsf{D}_\supset$
1. $P \Rightarrow P, Q$	assunção

É de destacar a simplicidade desta prova quando comparada com a prova nos sistemas de dedução natural.

Refutação de estados de prova

A correcção de um sistema de dedução orientado a sequentes é, normalmente, expresso em termos da refutação de sequentes.

Informalmente um sequente $\Gamma \Rightarrow \Delta$ é **refutado** quando é possível que todas as proposições $P \in \Gamma$ sejam *válidas* e todas as proposições $Q \in \Delta$ sejam *não válidas*.

Inversamente o sequente é **aceite** se, sempre que todos as $P \in \Gamma$ sejam *válidas*, alguma $Q \in \Delta$ é também *válida*.

A noção de validade por refutação pode ser agora aplicada a estados de prova e, em particular, a regras de inferência.

Considere-se um estado de prova definido por uma árvore

$$\dfrac{\begin{array}{ccc} R_1 & & R_n \\ \vdots & \ldots & \vdots \end{array}}{S}$$

14 Definição *O estado de prova é* ***correcto*** *quando toda eventual refutação de S implica a refutação de pelo menos uma das folhas R_i .*

7 Proposição *Se uma dedução é correcta e tem conclusão S então, necessáriamente, S não pode ser refutado (é aceite).*

Note-se que numa dedução é um estado de prova onde todas as folhas são fechadas por assunção; por isso nunca podem ser refutadas e consequentemente, sendo a dedução um estado de prova correcto, a conclusão também não pode ser refutada.

Os estados de prova ou são simples sequentes (folhas) ou são construídos por resolução a partir de outros estados de prova. Das definições conclui-se imediatamente

8 PROPOSIÇÃO *Uma folha é um estado de prova correcto. A resolução de dois estados de prova correctos é também um estado de prova correcto.*

Um estado de prova construído com regras de inferência correctas é sempre correcto.

A demonstração deste resultado é uma consequência simples das definições. O seguinte programa PROLOG implementa directamente essas definições e pode ser usado para verificar a correcção de um estado de prova.

```
correcto(U,[S]).
correcto(U,[S|_])  :- \+ refuta(U,_,S).
correcto(U,[C|P])  :-
  setof(X,refuta(U,X,C),Mc), setof(Y,refutaF(U,Y,P), Mf),
  sublist(Mc,Mf).
refutaF(U,M,D)     :- folhas(D,F), refuta(U,M,F).
```

O predicado `correcto` recebe como parâmetros o universo `U` dos símbolos proposicionais e a dedução e verifica a definição de dedução calculando o conjunto `Mc` dos modelos que refutam a conclusão e o conjunto `Mf` dos modelos que refutam uma das folhas.

Os predicados `folhas` e `refuta` são implementados separadamente e determinam as folhas de uma dedução e os modelos que refutam um dado sequente.

```
| ?- correcto([p], [ [] ==> [p] ]).
yes
| ?- correcto([p,q], [ [] ==> [p => q] , [[p] ==> [q]]  ]).
yes
| ?- correcto([p,q], [ [] ==> [p & q] , [[] ==> [q]] ]).
no
```

Correcção do Sistema LK + (CORTE)

Como consequência da proposição 8 basta que todas as regras de inferência sejam estados de prova correctos para que todo o estado de prova, construído de acordo com a definição 13, seja correcto.

Em particular uma dedução para a relação $\Gamma \vdash C$ (que tem $\Gamma \Rightarrow C$ como conclusão), sendo construída com regras de inferência correctas, tem uma conclusão necessariamente não refutável (proposição 7). Nessas circunstâncias verifica-se $\Gamma \models C$ e a relação de dedução é, portanto, correcta.

Em conclusão, para mostrar a correcção do sistema de dedução LK + (CORTE), basta provar a correcção de todas as suas regras de inferência.

4 TEOREMA *Todas as regras de inferência de* LK + (CORTE) *são estados de prova correctos.*

A demonstração deste resultado faz-se aplicando a definição 14 a cada uma das regras de inferência. Apenas como exemplo, ilustra-mos essa prova na regra

$$\frac{\Gamma, Q \Rightarrow \Delta \qquad \Gamma \Rightarrow P, \Delta}{\Gamma, P \supset Q \Rightarrow \Delta}\ (\mathsf{E}_{\supset})$$

Um modelo, que refute a conclusão, valida $\{\Gamma, \neg\Delta\}$ e valida $P \supset Q$; para tal, ou valida Q ou então valida $\neg P$.

Portanto o modelo valida $\{\Gamma, Q, \neg\Delta\}$ ou então valida $\{\Gamma, \neg P, \neg\Delta\}$; isto significa que refuta o sequente $\Gamma, Q \Rightarrow \Delta$ ou então refuta o sequente $\Gamma \Rightarrow P, \Delta$.

De acordo com a definição 14 isto implica que a regra $(\mathsf{E}_{\supset})$ é um estado de prova correcto.

Para as restantes regras procede-se de forma análoga.

Exemplos

Alguns exemplos de deduções simples no sistema LK+(CORTE)

$P \vee \neg P$

1. $\Rightarrow P \vee \neg P$	$D_\vee$
1. $\Rightarrow P, \neg P$	$D_\neg$
1. $P \Rightarrow P$	assunção

$P \vee P \supset P$

1. $\Rightarrow P \vee P \supset P$	$D_\supset$
1. $P \vee P \Rightarrow P$	$E_\vee$
1. $P \Rightarrow P$	assunção
2. $P \Rightarrow P$	assunção

$P \wedge R \supset Q \wedge R$	$[\, P \supset Q \,]$

1. $P \supset Q \Rightarrow P \wedge R \supset Q \wedge R$	$D_\supset$
1. $P \supset Q, P \wedge R \Rightarrow Q \wedge R$	$E_\wedge$
1. $P \supset Q, P, R \Rightarrow Q \wedge R$	$D_\wedge$
1. $P \supset Q, P, R \Rightarrow Q$	$E_\supset$
1. $Q, P, R \Rightarrow Q$	assunção
2. $P, R \Rightarrow P, Q$	assunção
2. $P \supset Q, P, R \Rightarrow R$	assunção

É óbvio, nestes exemplos, que cada dedução usa tantas resoluções com regras esquerdas ou direitas quantos os conectivos no sequente a provar. Por exemplo, a última dedução tem 4 conectivos e cada uma das resoluções elimina um dos conectivos.

Equivalência entre LK+(CORTE) e $\mathbf{Ns}_c$

O sistema LK+(CORTE) foi motivado pelo objectivo de construir um sistema de dedução equivalente à dedução natural (que justifique, portanto, as mesmas relações $\Gamma \vdash C$) e que tenha apenas regras que satisfaçam a propriedade da sub-fórmulas.

O sistema LK+(CORTE) aproxima-se destes objectivos mas é necessário mostrar que é equivalente à dedução natural. Para isso é necessário provar

9 PROPOSIÇÃO *Toda a regra do sistema* LK+(CORTE) *é uma consequência do sistema* $\mathbf{Ns}_c$ *e, vice-versa, toda a regra deste último é uma consequência do primeiro.*

Para construir provar, no sistema $\mathbf{Ns}_c$, regras de LK+(CORTE) é necessário representar sequentes generalizados $\Gamma \Rightarrow \Delta$ (onde o lado direito pode ser qualquer conjunto finito) em sequentes $\Gamma \Rightarrow C$ usados em $\mathbf{Ns}_c$. Para isso um sequente $\Gamma \Rightarrow \{\,\}$ representa-se por $\Gamma \Rightarrow \bot$ e um sequente $\Gamma \Rightarrow C, \Delta$ representa-se por $\Gamma, \neg\Delta \Rightarrow C$.

Por exemplo, para provar a regra

$$\frac{\Gamma, P \Rightarrow Q, \Delta}{\Gamma \Rightarrow P \supset Q, \Delta}\ (\mathsf{D}_{\supset})$$

usa-se a codificação para escrever essa regra da forma

$$\frac{\Gamma, \neg\Delta, P \Rightarrow Q}{\Gamma, \neg\Delta \Rightarrow P \supset Q}$$

onde se reconhece a regra de introdução de $\supset$ na dedução natural.

A segunda parte exige uma prova em LK+(CORTE) para cada regra de $\mathbf{Ns}_c$. Ilustra-mos o processo com a regra *Modus Ponens*.

$$\dfrac{\dfrac{\Gamma \Rightarrow P \quad \dfrac{}{Q \Rightarrow Q}\ (\mathsf{ass})}{\Gamma, P \supset Q \Rightarrow Q}\ (\mathsf{E}_{\supset}) \qquad \Delta \Rightarrow P \supset Q}{\Gamma, \Delta \Rightarrow Q}\ (\mathsf{corte})$$

Implementação em Prolog

Apresentamos agora um esboço[a] de um programa Prolog que constrói quadros de prova por propagação em retrocesso no sistema LK.

Dada a necessidade de representar teorias por listas, e dado que a manipulação de listas é sempre feita pela "cabeça" da lista, cada teoria é aqui representada por duas listas ligadas pelo símbolo `@`. A primeira lista contém as proposições antes da fórmula principal, enquanto a segunda contém a fórmula principal (à cabeça) e as proposições restantes.

O segundo conjunto de clausulas do predicado `lk` usa esta representação para transferir um candidato a fórmula principal, que não foi processado pelas duas clausulas anteriores, da segunda destas listas para a primeira.

```
deduc(P) :- lk( []@[] ==> [] @ [P]).
%
lk(S) :- assuncao(S),   print('assu'), print(S), nl.
lk(C) :- regra(R,C,Prems), print(R), print(C), nl, lks(Prems).
%
lk( Ge @ [P|Gd] ==> De @ Dd)  :- lk([P|Ge] @ Gd ==> De @ Dd).
lk( Ge @ Gd ==> De @ [P|Dd])  :- lk(Ge @ Gd ==> [P|De] @ Dd).
%
lks([],_).
lks([S|R],N) :- lk(S,N), lks(R,N).
%
assuncao(Ge@Gd ==> De@Dd) :-
   append(Ge,Gd,G), append(De,Dd,D), member(C,G), member(C,D).
regra(conE, Ge@[P & Q|Gd] ==> De@Dd , [Ge@[P,Q|Gd] ==> De@Dd]).
regra(impD, Ge@Gd ==> De@[P => Q|Dd], [Ge@[P|Gd] ==> De@[Q|Dd] ]).
```

Estão aqui apresentadas apenas duas das regras de inferência. As restantes representam-se de forma análoga e estão disponíveis no texto completo do programa.

(*continua*)

[a] O texto completo deste programa está disponível no `URL` da disciplina.

Exemplos de deduções construídas por este programa:

(a) $((P \supset Q) \supset P) \supset P$

```
| ?- deduc(((p => q) => p) => p).

impD []==>[((p=>q)=>p)=>p]
impE  [(p=>q)=>p]==>[p]
assu   [p]==>[p]
impD   []==>[p=>q,p]
assu    [p]==>[q,p]

yes
```

(b) $(P \supset \neg Q) \supset (Q \supset \neg P)$

```
| ?- deduc((p => -(q)) => (q => -(p))).

impD []==>[(p=> -(q))=>q=> -(p)]
impE  [p=> -(q)]==>[q=> -(p)]
negE   [-(q)]==>[q=> -(p)]
impD    []==>[q,q=> -(p)]
assu     [q]==>[q,-(p)]
impD   []==>[p,q=> -(p)]
negD    [q]==>[p,-(p)]
assu     [p,q]==>[p]

yes
```

O cálculo de sequentes LJ

.

O sistema **LJ** é a versão de LK usando "sequentes intuicionistas"; isto é, sequentes em que o lado direito tem, quanto muito, uma fórmula.

As noções de dedução e estado de prova são as mesmas que em LK e apenas muda o conjunto de refras de inferência.

Regra do Corte

$$\frac{\Gamma, C \Rightarrow \Delta \qquad \Gamma' \Rightarrow C}{\Gamma, \Gamma' \Rightarrow \Delta}\ (\text{corte})$$

Regras "Direitas" (conclusões)

$$\frac{\Gamma \Rightarrow P \qquad \Gamma \Rightarrow Q}{\Gamma \Rightarrow P \wedge Q}\ (\mathsf{D}_\wedge) \qquad \frac{\Gamma \Rightarrow P}{\Gamma \Rightarrow P \vee Q}\ (\mathsf{D}_{\vee 1}) \qquad \frac{\Gamma \Rightarrow Q}{\Gamma \Rightarrow P \vee Q}\ (\mathsf{D}_{\vee 2})$$

$$\frac{\Gamma, P \Rightarrow Q}{\Gamma \Rightarrow P \supset Q}\ (\mathsf{D}_\supset) \qquad \frac{\Gamma \Rightarrow}{\Gamma \Rightarrow \bot}\ (\mathsf{D}_\bot)$$

Regras "Esquerdas" (assunções)

$$\frac{\Gamma, P, Q \Rightarrow \Delta}{\Gamma, P \wedge Q \Rightarrow \Delta}\ (\mathsf{E}_\wedge) \qquad \frac{\Gamma, P \Rightarrow \Delta \qquad \Gamma, Q \Rightarrow \Delta}{\Gamma, P \vee Q \Rightarrow \Delta}\ (\mathsf{E}_\vee)$$

$$\frac{\Gamma, Q \Rightarrow \Delta \qquad \Gamma, P \supset Q \Rightarrow P}{\Gamma, P \supset Q \Rightarrow \Delta}\ (\mathsf{E}_\supset) \qquad \frac{}{\Gamma, \bot \Rightarrow \Delta}\ (\mathsf{E}_\bot)$$

Comentários

(i) Note-se que as regras da negação foram substituídas por regras "esquerda" e "direita" para o absurdo $\perp$. A negação é definida por abreviatura

$$\neg A \quad \equiv \quad A \supset \perp$$

A regra direita afirma que um lado direito vazio é equivalente ao absurdo. A regra esquerda diz que qualquer sequente com uma assunção absurda não pode ser refutado.

Esta regra esquerda, porque não tem quaisquer premissas, constitui uma forma legítima de fechar estados de prova; no sistema LJ as provas podem ser fechadas por assunção (como em LK) ou por resolução com a regra $\mathsf{E}_{\perp}$.

(ii) A regra "esquerda" da implicação contém uma premissa um tanto inesperada: a fórmula principal $P \supset Q$ mantém-se no lado esquerdo de uma das premissas.

(iii) Como os lados direitos dos sequentes têm quanto muito uma fórmula, a regra $\mathsf{D}_{\vee}$ tem de ser decomposta em duas regras.

Exemplo

$A \vee B \supset C$	$[\, A \supset C, B \supset C \,]$
1. $A \supset C, B \supset C \Rightarrow A \vee B \supset C$	$\mathsf{D}_{\supset}$
1. $A \supset C, B \supset C, A \vee B \Rightarrow C$	$\mathsf{E}_{\vee}$
1. $A \supset C, B \supset C, A \Rightarrow C$	$\mathsf{E}_{\supset}$
1. $B \supset C, A, C \Rightarrow C$	assunção
2. $B \supset C, A, A \supset C \Rightarrow A$	assunção
2. $A \supset C, B \supset C, B \Rightarrow C$	$\mathsf{E}_{\supset}$
1. $A \supset C, B, C \Rightarrow C$	assunção
2. $A \supset C, B, B \supset C \Rightarrow B$	assunção