Leão, Jorge Lopes de Souza
Lógica para computação (Rio de Janeiro) 2004, 2013, 2016
IV , 92 + 39 p, 29,7cm (COPPE/UFRJ)
Inclue referências bibliográficas e apêndices.
I.COPPE/UFRJ II.Título
1. Lógica
2. Computação
3. PROLOG
4. ASP

Prefácio

Este resumo de lógica está organizado da seguinte maneira: o capítulo 1, Lógica proposicional, apresenta tanto a linguagem proposicional quanto os sistemas de dedução correspondentes. O capítulo 2, Linguagens de Primeira Ordem, apresenta somente a sintaxe e a semântica das linguagens de primeira ordem. O capítulo 3, Máquinas e linguagens, apresenta os conceitos de máquinas computacionais, problemas de decisão e a sua relação com as linguagens. O capítulo 4, Cláusulas de 1ª Ordem e Resolução, apresenta uma família de linguagens de primeira ordem que pode ser vista como uma simplificação sintática da linguagem de fórmulas de primeira ordem. Além disto, ele apresenta o sistema axiomático da resolução.

Com a finalidade de manter uma notação homogênea e ajudar uma recordação de conceitos mais básicos, um apêndice sobre a teoria dos conjuntos foi acrescentado.

JLSL, abril de 2016.

Conteúdo

1. LÓGICA PROPOSICIONAL

Uma lógica baseia-se em uma linguagem e em um sistema de deduções. Uma das abordagens clássicas para o estudo das linguagens é definir inicialmente um vocabulário (a sua parte léxica), depois definir a sintaxe e em seguida a semântica. A definição de um sistema formal de deduções completa a definição de uma lógica.

1.1 Vocabulário

O vocabulário de uma lógica proposicional é formado pelos seguintes conjuntos:

símbolos proposicionais	:	S_P	= { p, q, r, ... }
símbolos dos conectivos	:	S_C	= { not , or }
símbolos de pontuação	:	S_{PO}	= { (,) }

Os símbolos proposicionais também são chamados de variáveis proposicionais.

O conjunto de símbolos proposicionais deve ser enumerável e, além disto, os três conjuntos devem ser disjuntos:

$$S_P \cap S_C = S_P \cap S_{PO} = S_C \cap S_{PO} = \varnothing$$

Sua união forma o vocabulário A da linguagem em questão:

$$A = S_P \cup S_C \cup S_{PO}$$

Os elementos de A também são chamados lexemas (*tokens*, em inglês).

Um símbolo proposicional, como **p** por exemplo, terá, mais tarde na definição da semântica, um significado tal como "esta chovendo" ou "n maior que zero".

Embora haja a tradição de se utilizar simples letras para os símbolos proposicionais, também é possível utilizar-se identificadores autoexplicativos como é usual nas linguagens de programação. Assim, poderíamos ter os símbolos proposicionais **estaChovendo** ou **n_maior_que_zero**. Justamente devido a esta tradição de usar-se simplesmente letras como símbolos proposicionais, também se usa o termo **alfabeto** como sinônimo de **vocabulário**, o que entretanto não corresponde ao significado utilizado em outros contextos.

1.2 Sintaxe

Vamos agora definir a sintaxe, que é o conjunto de regras que define o que é uma frase legal, bem escrita ou bem formada.

As frases da lógica são chamadas fórmulas e elas são construídas com os elementos de um dado vocabulário A. Diremos então que elas são ***fórmulas sobre A***.

A definição de fórmula é dada de maneira indutiva por:

Def. 1:
f é uma fórmula sobre A
sse
(caso base) f é um símbolo proposicional de A ou
(passo de indução) se P e Q são fórmulas sobre A,
então (P) , (not P) e (P or Q) são fórmulas sobre A.

Def. 2:
Uma linguagem proposicional sobre um dado vocabulário A, denotada por $L_0(A)$, é o conjunto de todas as fórmulas sobre A.

Neste ponto é importante esclarecer que, para falar sobre a linguagem proposicional, precisamos usar uma outra linguagem. A linguagem que usamos para falar da *linguagem objeto* (neste caso a linguagem proposicional) é chamada de *metalinguagem*. Mesmo querendo ser preciso e não ambíguo, não podemos definir a metalinguagem de uma maneira formal, com vocabulário, sintaxe e semântica bem definidas, pois precisaríamos de uma *metametalinguagem*. A solução encontrada é usarmos como metalinguagem um fragmento de uma linguagem natural acrescida de uma notação no estilo matemático (no nosso caso um português *matematizado*), para as quais existe um consenso, tanto acerca da sintaxe como acerca da semântica. Vale a pena comentar que este tipo de consenso é a base para todo o desenvolvimento científico, não só da matemática, como também das ciências naturais. O estudo das possibilidades do uso de tal metalinguagem ultrapassa a própria matemática, caindo no domínio da metamatemática [...REF...].

No caso da definição de fórmula, podemos observar que os símbolos f, P e Q, assim como o símbolo " $\equiv_{def}$ " (*idêntico por definição*), pertencem à metalinguagem, assim como o próprio texto da definição.

Utilizaremos os seguintes símbolos adicionais como abreviações:

(P and Q) $\equiv_{def}$ (not ((not P) or (not Q)))
(P implies Q) $\equiv_{def}$ (Q if P) $\equiv_{def}$ ((not P) or Q)
(P iff Q) $\equiv_{def}$ (P implies Q) and (Q implies P)

Para simplificar a manipulação das fórmulas, utilizaremos as seguintes convenções para a omissão de parênteses, em ordem decrescente de prioridade:

1) A negação aplica-se à menor fórmula possível, se não houver parênteses:
(not A or B)
é equivalente a
((not A) or B)

2) A conjunção e a disjunção aplicam-se à menor fórmula possível, se não houver parênteses :
(A and not B implies C)
é equivalente a
((A and (not B)) implies C)

3) Quando um conectivo é usado repetidamente, o agrupamento é feito pela direita:

(A implies B implies C)
é equivalente a
(A implies (B implies C))

Além disto, os parênteses exteriores são opcionais.

1.3 Semântica

Vamos agora definir a semântica ou significado das linguagens proposicionais. As fórmulas das linguagens proposicionais só têm como significado um de dois valores, que são normalmente chamados de V e F, T e F, true e false, 1 e 0, ou verdadeiro e falso.

É importante observar que estes dois símbolos não fazem parte do vocabulário da linguagem, mesmo que, por coincidência, existam outros símbolos no vocabulário com os mesmos caracteres.

O significado das fórmulas vai ser dado então por duas funções que vão mapear os símbolos proposicionais, e depois as fórmulas, no conjunto de valores-verdade Bool = { true , false }.

A primeira função é a atribuição de valores-verdade aos símbolos proposicionais, dada por:

$$v : S_P \longrightarrow \text{Bool}$$

Esta função é arbitrária e para cada escolha diferente teremos, em princípio, um significado diferente para as fórmulas.

Intuitivamente, cada atribuição corresponde a uma situação sendo descrita pela linguagem proposicional (estado de coisas ou simplesmente estado).

A segunda função é a avaliação das fórmulas, induzida por v, e dada por:

$$V[v] : L_0(A) \longrightarrow \text{Bool}$$

onde,

$$V[v](P) = v(P) \qquad \text{se } P \text{ é uma fórmula atômica}$$

$$V[v](\text{not } P) = \begin{cases} \text{true} & \text{se } V[v](P) = \text{false} \\ \text{false} & \text{se } V[v](P) = \text{true} \end{cases}$$

$$V[v](P \text{ or } Q) = \begin{cases} \text{false} & \text{se } V[v](P) = V[v](Q) = \text{false} \\ \text{true} & \text{em caso contrário} \end{cases}$$

Def. 3:
Uma fórmula f é **satisfatível**
sse
para alguma atribuição v, V[v](f) = true.

O mesmo conceito pode ser expresso, ao invés da função de avaliação V, através da relação de satisfação.

Def. 4:
v satisfaz f , e escreve-se $v \models f$
sse
V[v](f) = true

Obs: $\models \subseteq ((S_P \to Bool) \times L_0(A))$

Def. 5:
Uma fórmula f é **válida** (ou uma **tautologia**)
sse
para toda atribuição v, V[v](f) = true.
Neste caso, escreve-se:

$\models f$

Def. 6:
Uma fórmula é **contraditória** (ou uma **contradição**, ou **insatisfatível**)
sse
para toda atribuição v, V[v](f) = false.

Prop. 1:
$\models f$
sse
(not f) é insatisfatível.

Dem:
($\to$)
§1. Considerando uma atribuição v qualquer e
a definição de fórmula válida,
conclui-se que
$v \models f$.
§2. Considerando a definição da negação e
o item 1,
conclui-se que
$v \not\models$ (not f).
§3. Considerando a definição de insatisfatível,
conclui-se que,
(not f) é insatisfatível.

($\leftarrow$)
§1. Considerando a definição de insatisfatível e
uma atribuição v qualquer,

conclui-se que
$v \not\models (\text{not } f)$.

§2. Considerando a definição de negação
conclui-se que
$v \models f$.

§3. Considerando o item 1 e
a definição de validade,
conclui-se que
$\models f$.

Def. 7:
Um conjunto de fórmulas **P** = { f_1 f_2 f_3 ... }, não vazio, é satisfatível
sse
para alguma atribuição v e para toda f_i , com i = 1, 2, 3 , ... , V[v](f_i) = true.

Se o conjunto de fórmulas **P** = ∅, então P é satisfatível.

O conjunto **P** é insatisfatível em caso contrário.

Para justificar a postulação de ∅ como satisfatível, basta observar que supondo um conjunto satisfatível, não vazio, de fórmulas P, o conjunto P∪∅ só será satisfatível se ∅ for satisfatível.

Def. 8:
Dados um conjunto de fórmulas **P** = { f_1 f_2 f_3 ... } e uma fórmula f,
P implica logicamente em f
sse
para toda atribuição v,
se v satisfaz **P** então v satisfaz f.

Neste caso, também diz-se que f é conseqüência lógica de **P** e escreve-se:

$\mathbf{P} \models f$

Caso contrário, escreve-se:

$\mathbf{P} \not\models f$

Prop. 2:
Dados um conjunto de fórmulas $\mathbf{P} = \{ f_1, f_2, f_3, \ldots \}$ e uma fórmula f,
$\mathbf{P} \models f$
sse
$\mathbf{P} \cup \{ (\text{not } f) \}$ é insatisfatível.

Dem:
(→)
(←)

Corolário 1:

As tautologias são conseqüência lógica do conjunto vazio, ou seja:

$$\varnothing \models f \text{ sse } \models f$$

Dem:
($\rightarrow$) Pela definição da implicação lógica ($\models$), como $\varnothing$ é satisfeito para todas as interpretações, então f deve ser verdadeira para todas as interpretações. Logo f é uma fórmula válida.
($\leftarrow$) Se f é uma fórmula válida, então f é verdadeira para todas as interpretações. Logo f é consequência lógica de qualquer conjunto de fórmulas.

Neste caso, verificamos uma sugestiva coincidência das notações de satisfação e implicação lógica.

Em vista deste corolário, e considerando a postulação da satisfatibilidade do conjunto de fórmulas vazio, passaremos a utilizar a expressão:
- O conjunto vazio de fórmulas é (trivialmente) satisfatível.

Com isto, evitamos uma frase que poderia sugerir um paradoxo:
- As fórmulas do conjunto vazio de fórmulas são satisfeitas por qualquer atribuição.

1.4 Axiomática

Def. 9:
Um sistema formal é um par (L,D), onde L é uma família de linguagens e D é um conjunto de sequências de fórmulas em uma mesma linguagem chamadas deduções. Neste capítulo consideraremos L_0 a família das linguagens proposicionais, que diferem umas das outras apenas pelos símbolos proposicionais, quer dizer, diferem pelo vocabulário A.

Def. 10:
Um sistema axiomático Ax é um sistema formal cujas deduções são formadas da maneira descrita a seguir:
- seja X um conjunto de fórmulas, possivelmente vazio, finito ou infinito, chamadas de axiomas;
- seja P um conjunto de fórmulas chamadas de hipóteses, ou conjunto de partida;
- seja R = { r_i } um conjunto de funções que mapeiam fórmulas em uma fórmula, isto é:

$$r_i : L_0(A)^n \rightarrow L_0(A)$$

onde, para cada r_i temos n, um número inteiro não negativo, que é a aridade da função.

Estas transformações são chamadas regras de inferência e as fórmulas do domínio são chamadas de premissas.

Uma dedução de uma fórmula f a partir de **P** é uma seqüência $f_1\ f_2\ f_3\ \ldots\ f_n$ de fórmulas tal que f= f_n e toda f_k , para $1 \leq k \leq n$, é ou um axioma, ou uma fórmula de P, ou é obtida pela aplicação de uma das regras de inferência a fórmulas f_j da dedução para $i < k$.

A fórmula f é chamada de um teorema em Ax a partir de **P**. Neste caso denota-se:

$\mathbf{P} \vdash f$

Diz-se também que f é um teorema em Ax sse f é um teorema em Ax a partir do conjunto de hipóteses vazio. Neste caso denota-se:

$\vdash f$

Def. 11:
Um sistema axiomático é **consistente** (“sound”)
sse
para toda fórmula f,
se $\vdash f$ então $\models f$

Prop. 3:
Um sistema axiomático Ax é consistente
sse
(a) todos os esquemas de axiomas são válidos,
(b) todas as regras de inferência preservam validade.

O termo *sistema axiomático consistente* possui uma segunda interpretação:
Def. 12:
Um sistema axiomático Ax é consistente
sse
para toda fórmula f,
se $\vdash f$ então $\nvdash$ (not f)
e
se $\vdash$ (not f) então $\nvdash f$

Os termos consistente e inconsistente também são utilizados na seguinte proposição com um significado semântico.

Prop. 4:
Dados um conjunto de fórmulas $P = \{ f_1, f_2, f_3, \ldots \}$, com **P** satisfatível, e uma fórmula f,
sse
se $\mathbf{P} \models f$ então $\mathbf{P} \not\models$ (not f)
e
se $\mathbf{P} \models$ (not f) então $\mathbf{P} \not\models f$
Neste caso, também dizemos que **P** é consistente.

Dem:
($\rightarrow$)
($\leftarrow$)

Def. 13:
Um sistema axiomático é **correto** (“strongly sound”)
sse

para toda fórmula f e todo conjunto de hipóteses **P**,
se $\mathbf{P} \vdash f$ então $\mathbf{P} \models f$

Def. 14:
Um sistema axiomático é **completo**
sse
para toda fórmula f e todo conjunto de hipóteses **P**,
se $\mathbf{P} \models f$ então $\mathbf{P} \vdash f$

Exemplo:

Considere a família de linguagens de ordem zero $L_0(A)$ (para todos os vocabulários A).
Considere o conjunto de axiomas X vazio.
Considere um conjunto de hipóteses (conjunto de partida) P não vazio, tal que P não seja insatisfatível.
Considere o conjunto de regras de inferência R composto apenas pela seguinte regra:

$$MP \equiv_{def} \frac{a, a \supset b}{b} = \frac{a, \neg a \vee b}{b}$$

Verifique se o sistema axiomático assim definido é correto e completo.
Obs: O nome escolhido para a regra significa "Modus Ponens", o seu nome histórico (Segundo a Wikipedia, ***modus ponendo ponens*** do Latin "a maneira que afirma afirmando").

Exemplo:

Considere a família de linguagens de ordem zero $L_0(A)$ (para todos os vocabulários A).
Considere o conjunto de axiomas X vazio.
Considere um conjunto de hipóteses (conjunto de partida) P não vazio, tal que P não seja insatisfatível.
Considere o conjunto de regras de inferência R composto apenas pela seguinte regra:

$$\mathrm{RE} \equiv_{\mathrm{def}} \frac{a \vee b, \neg a \vee c}{b \vee c}$$

Verifique se o sistema axiomático assim definido é correto e completo.
Obs: O nome escolhido para a regra significa "Resolução" e é uma alusão à regra de mesmo nome usada na lógica de ordem um.

2. LINGUAGENS DE PRIMEIRA ORDEM

As linguagens proposicionais têm limitações na sua expressividade, o que levou ao desenvolvimento das linguagens de primeira ordem.

2.1. Vocabulário

A definição de uma linguagem de predicados, ou de primeira ordem, é feita de maneira semelhante, isto é, começando pela definição do seu vocabulário:

constantes	: S_{CTE}	= { a, b, c, ... }
funcionais n-ários	: S_{FUNC}	= { f, g, h, ... }
predicativos n-ários	: S_{PRED}	= { p, q, r, ... }
variáveis	: S_V	= { x, y, z, ... }
quantificadores	: S_Q	= { forSome }
conectivos	: S_C	= { not , or }
pontuação	: S_{PO}	= { (,) , [,] , ,, }

Os conjuntos S_{CTE}, S_{FUNC}, S_{PRED} e S_V são enumeráveis. Como anteriormente, temos que estes sete conjuntos devem ser disjuntos e o vocabulário A é dado pela união destes. As constantes também podem ser vistas como símbolos funcionários nulários, isto é, de aridade zero.

2.2. Sintaxe

A sintaxe da lógica de primeira ordem é dada em duas partes.
Primeiro, vamos definir os termos e depois as fórmulas sobre A.

Termos:

(i) Toda variável de A é um termo sobre A.
(ii) Toda constante de A é um termo sobre A.
(iii) Se t_1, t_2, ... e t_n são termos sobre A e f é um símbolo funcional n-ário de A, então $f(t_1, t_2, ..., t_n)$ é um termo sobre A.
(iii) nenhuma outra cadeia é um termo sobre A.

Os termos são usados para dar nome, ou denotar, objetos de um universo de discurso, como veremos na semântica, enquanto as fórmulas são usadas para fazer afirmativas sobre estes objetos.

Chamaremos $T_1(A)$ o conjunto de todos os termos de primeira ordem sobre o vocabulário A.

O conjunto das fórmulas é dado por:

(i) Se t_1, t_2, ... e t_n são termos sobre A e p é um símbolo predicativo n-ário de A, então $p(t_1, t_2, ..., t_n)$ é uma fórmula (atômica) sobre A.

(ii) Se P e Q são fórmulas sobre A, então as seguintes cadeias também são fórmulas sobre A:

(P) ,
(not P) ,
(P or Q) .

(iii) Se P é uma fórmula sobre A e x é uma variável de A, então a seguinte cadeia também é uma fórmula sobre A:

(forSome[x] P)

(iv) nenhuma outra cadeia é uma fórmula sobre A.

O conjunto de todas as fórmulas e termos sobre A, denotado por $L_1(A)$, é a linguagem de primeira ordem sobre A.

Continuaremos a utilizar as convenções de abreviação de conectivos e de parenteses usadas na lógica de ordem zero, adicionadas da seguinte abreviação:

(forAll[x] P) $\equiv_{def}$ (not (forSome[x] (not P)))

Também estenderemos, para os dois quantificadores, a regra de abreviação de parênteses que diz que quando um conectivo é usado repetidamente o agrupamento é feito pela direita. A fórmula (forAll[x] (forAll[y] ... (forAll[z] P) ...) pode então ser escrita sem os parênteses como

forAll[x]forAll[y]. . . forAll[z] P

ou

forAll[x,y, . . . ,z] P.

Uma outra notação muito conhecida para os conectivos e quantificadores é:

not	$\neg$	, a negação;
or	$\vee$	, a disjunção;
and	$\wedge$	, a conjunção;
implies	$\supset$	, a implicação (material);
iff	$\equiv$	, a equivalência;
forSome[...]	$\exists$[...]	, o quantificador existencial (também se usa ?[...]);
forAll[...]	$\forall$[...]	, o quantificador universal (também se usa ![...]).

Nesta notação temos: ($\exists$x P) $\equiv_{def}$ (forSome[x] P)

Os colchetes junto aos quantificadores são opcionais.

Def:
Diz-se que uma ocorrência de uma variável x numa fórmula P é ligada
sse
esta ocorre numa subfórmula de P da forma (forAll[x] Q) ou (forSome[x] Q).
Ela é livre em caso contrário.
Diremos então que a subfórmula Q é o escopo desta quantificação.

Def:
Uma fórmula f de $L_1(A)$ é uma sentença
sse
f não possui variáveis livres.

Preferiremos utilizar as fórmulas com uma renomeação canônica de suas variáveis, tal que:
(i) as variáveis são renomeadas para x_1, x_2, x_3, ...
(ii) toda subfórmula da forma (forSome[x_i] Q) não contém ocorrências ligadas de x_i em Q,
(iii) para toda variável x_i de uma fórmula, as ocorrências de x_i nesta fórmula são ou todas ligadas ou todas livres, exclusivamente.
(iv) se existirem ocorrências ligadas de x_i em uma fórmula, elas ocorrem todas numa única subfórmula da forma (forSome[x_i] Q).

2.3. Semântica

A semântica de uma linguagem de primeira ordem vai ser dada por um universo de discurso U e por três conjuntos de funções:

a estrutura,
a atribuição de valores às variáveis, e
a avaliação dos termos e das fórmulas.

Uma estrutura E para o vocabulário A sobre o universo de discurso U é um funcional dado por:

(i) $E_{CTE} : S_{CTE} \longrightarrow U$

(ii) $E_{FUNC} : S_{FUNC} \longrightarrow (U^n \longrightarrow U)$, onde n é a aridade de cada símbolo funcional de S_{FUNC}.

(iii) $E_{PRED} : S_{PRED} \longrightarrow 2^{U^n}$, onde n é a aridade de cada símbolo predicativo de S_{PRED}.

O par (U,E), composto pelo universo de discurso U e pela estrutura E é também chamado de **interpretação**.

Uma atribuição de valores às variáveis, também chamada de estado, é uma função:

$$s : S_V \longrightarrow U$$

É oportuno dizer que esta é uma função parcial, pois somente as variáveis livres de uma fórmula precisarão de uma atribuição para a avaliação dos termos e, consequentemente, da própria fórmula.
A avaliação dos termos, induzida pela interpretação I e pela atribuição s é uma função:

$$v[I,s] : T_1(A) \longrightarrow U$$

Esta função é definida por:

(i) $v[I,s](x) = s(x)$, se x pertence a S_V

(ii) $v[I,s](c) = E_{CTE}(c)$, se c pertence a S_{CTE}

(iii) $v[I,s](f(t_1, t_2, ..., t_n)) = E_{FUNC}(f)(v[I,s](t_1) , ... , v[I,s](t_n))$, se f pertence a S_F e $t_1, t_2, ..., t_n$ são termos.

A definição de variância de uma função irá facilitar a definição da avaliação das fórmulas.

Def:
Sejam f e f' duas funções de C em D (não necessariamente totais): $f , f' : C \rightarrow D$ e x um elemento de C.
f' é uma variância x de f , ou **$f' =_x f$** (**f'** é igual a **f** a menos de x) ,
sse
para todo y, $y \in C-\{x\}$, $f'(y) = f(y)$.

A avaliação das fórmulas, induzida pela interpretação I e pela atribuição s, é uma função:

$V[I,s] : L_1(A) - T_1(A) \longrightarrow Bool$

Esta função é definida como dado abaixo, supondo que P e Q são fórmulas e p é um símbolo predicativo n-ário.

(i) $V[I,s](p(t_1, t_2, ..., t_n)) = true$, sse $(u_1, u_2, ..., u_n) \in I_P(p)$, onde $u_i = v[I,s](t_i)$, para i=1,...,n ;

(ii) V[I,s](not P) = true , sse V[I,s](P) = false ;

(iii) V[I,s](P or Q) = false , sse V[I,s](P) = V[I,s](Q) = false

(iv) V[I,s](forSome[x] P) = true , sse V[I,r](P) = true para alguma atribuição r tal que $r =_x s$.

Uma notação alternativa apresenta a função de avaliação das fórmulas na forma de uma relação de satisfação:

$I,s \vDash P$, sse V[I,s](P)=true

e

$I,s \nvDash P$, sse V[I,s](P)=false

Neste caso diz-se que P é **verdadeira** para I e s , ou que I e s **satisfazem** P. P é falsa, ou não é satisfeita, em caso contrário.

Também se poderia definir a avaliação das fórmulas, de uma forma mais compacta, por:

$I,s \vDash p(t1, t2, ..., tn)$ sse $(u1, u2, ..., un) \in IP(p)$, onde $ui = v[I,s](ti)$, para i=1,...,n ;

$I,s \vDash$ not P sse $I,s \nvDash P$

$I,s \vDash P$ or Q sse $I,s \vDash P$ ou $I,s \vDash Q$

I,s ⊨ forSome[x] P sse I,s ⊨ P para alguma atribuição r tal que $r =_x s$.

Definições:

- P é **válida em I** (escreve-se: I ⊨ P)
sse
para toda atribuição s , I,s ⊨ P.

- P é **válida** (escreve-se: ⊨ P)
sse
para toda interpretação I e para toda atribuição s , I,s ⊨ P.

Obs: Os conceitos de verdade e satisfatibilidade para I e s , para uma única fórmula, coincidem. Não será o caso quando considerarmos conjuntos de fórmulas, imediatamente a seguir.

Definições:

Seja agora Q um conjunto de fórmulas de $L_1(A)$.

- I **satisfaz** Q **para s**
(e neste caso escreve-se I,s ⊨ Q)
sse
para toda q∈Q, I,s ⊨ q.

- I **satisfaz** Q
sse
para alguma s , I,s ⊨ Q.

- Q é **satisfatível**
sse
para alguma I , para alguma s , I,s ⊨ Q.

- Q é **insatisfatível**
sse
Q não é satisfatível.

- f **é conseqüência lógica de** Q,
ou Q **implica logicamente em** f,
(e neste caso escreve-se Q ⊨ f, usando-se o mesmo símbolo anterior)
sse
para toda I e para toda s , se I,s ⊨ Q então I,s ⊨ f.

- Uma interpretação I é um **modelo** para um conjunto de sentenças S
sse
I ⊨ S.

Proposição:

Sejam: A um vocabulário, f uma fórmula de $L_1(A)$ e P um conjunto de fórmulas de $L_1(A)$.

a) f é válida sse (not f) é insatisfatível (Escreve-se $\models f$).
b) $P \models f$ sse $P \cup \{(\text{not } f)\}$ é insatisfatível.

Corolário:

As fórmulas válidas são consequência lógica do conjunto vazio, ou seja
$\varnothing \models f$ sse $\models f$.

Obs: O mesmo símbolo $\models$ é usado polimorficamente, isto é, com dois significados: satisfação ou implicação lógica.

3. TEORIAS DE PRIMEIRA ORDEM

4. MÁQUINAS DE TURING

O conceito de algoritmo é intimamente ligado ao conceito de computabilidade.

Dizendo de uma maneira coloquial, um algoritmo é uma receita finita, um conjunto de instruções que uma pessoa pode executar sem usar nenhuma forma especial de inteligência, ou que uma máquina apropriada também possa fazer. A receita é constituída de um número finito de passos que são executados um após o outro, em algum tipo de sequência, determinística ou não, e que podem conter instruções para realizar testes e repetições. Estas repetições podem ocorrer um número finito ou infinito de vezes. A receita também pode possuir um conjunto inicial finito de dados que podem ser consultados e modificados. Ela também deve possuir algum tipo de memória de cálculo onde resultados podem guardados, consultados e modificados. Esta memória deverá sempre finita durante toda a execução da receita, mas ela é ilimitada. Como as repetições podem ocorrer um número ilimitado de vezes, diz-se que a receita pode terminar ou não.

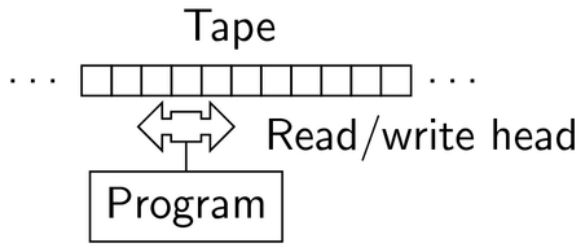
Uma versão mais formalizada deste conceito é a máquina de Turing. Outras versões formalizadas do mesmo conceito são as funções μ-recursivas, o cálculo lambda e várias outras máquinas e linguagens, incluindo a programação em lógica. Ao longo das décadas que se seguiram a partir de 1936, todas estas formulações alternativas mostraram-se equivalentes.

Uma máquina de Turing é composta pelos seguintes elementos:

a) Um vocabulário Γ, finito, não vazio de símbolos. Dentre estes símbolos, há um símbolo especial, que pode ser chamado, por exemplo, de λ, ou vazio.
b) Uma memória sequencial de células que armazenam cada uma uma ocorrência de um símbolo do vocabulário acima, eventualmente com repetições. Esta memória de células é infinita, porém durante todo o funcionamento da máquina, o único símbolo que pode ocorrer um número infinito de vezes na memória é o símbolo λ. Esta memória é conhecida como fita. Também existe um ponteiro que indica qual é a posição de memória da qual vai ser lido um símbolo ou onde vai ser escrito um símbolo. Este ponteiro é conhecido como a "cabeça de leitura e escrita".
c) Uma máquina de estados finita. Cada transição desta máquina tem uma etiqueta composta de uma condição e uma ação. As condições correspondem aos símbolos do vocabulário que são lidos pela cabeça de leitura na posição em que ela estiver. Existem portanto #Γ transições de saída de cada estado. As ações de qualquer

transição podem ser: (i)escrever um símbolo na fita, na posição atual da cabeça ou (ii)mover a cabeça uma posição para a esquerda ou (iii)mover a cabeça uma posição para a direita.

d) As condições iniciais da máquina, que são: o conteúdo da fita, a posição da cabeça na fita e o estado inicial da máquina de estados finita.



Funções computáveis e funções não computáveis.

Para ver um pouco mais de detalhes:
Barker-Plummer, David, "Turing Machines", *The Stanford Encyclopedia of Philosophy* (Spring 2016 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/spr2016/entries/turing-machine/>.

5. PROBLEMAS DE DECISÃO

6. ÁRVORES SEMÂNTICAS BINÁRIAS

7. CLÁUSULAS DE PRIMEIRA ORDEM E RESOLUÇÃO

8. CLÁUSULAS DEFINIDAS POSITIVAS

9. PROGRAMAÇÃO EM LÓGICA

10. A NEGAÇÃO

11. PROLOG

12. ANSWER SET PROGRAMMING