Lógica Computacional

DCC/FCUP

2020/21

Funcionamento da disciplina

- Docentes:
  - Teóricas: Sabine Broda
  - Práticas: Sabine Broda e Rita Ribeiro
- Avaliação:
  - Primeiro teste (PT) + Exame (E)

$$Final = \frac{1}{2}PT + \frac{1}{2}E^{1}$$

  - Recurso cotado para 20 valores.
  - Data do primeiro teste: a definir.

[1] $PT,\ E \geq 6$ e $Final \geq 9.5$

Alguma Bibliografia

- *"Logic in Computer Science"*, Michael Huth , 1962.
- *"Language, proof, and logic"*, Jon Barwise, 1999.
- *"Essentials of logic programming"*, Christopher John Hogger, 1990.
- *"Rigorous software development"*, José Bacelar Almeida et al., 2011.
- *"Mathematical Logic for Computer Science"*. Mordechai Ben-Ari 2001.

Objectivos

Introdução à lógica matemática numa perspectiva computacional.

- Noções básicas do raciocínio lógico e utilização correcta dos sistemas dedutivos
- Relação entre semântica e sistemas dedutivos: integridade e completude (o que é válido é deduzível; o que é deduzível é válido)...
- Lógica proposicional e Lógica de primeira ordem
- Axiomatizações
- Breve introdução à Programação em Lógica (base da linguagem de programação `prolog`).

Programa

1. Lógica proposicional
   - sintaxe; semântica (tautologias, satisfazibilidade, validade); formas normais; sistema dedutivo de dedução natural, *DN*; Completude e integridade do sistema *DN*; outros sistemas dedutivos (Hilbert, tableaux, resolução); decidibilidade e algoritmos de satisfazibilidade
2. Lógica de primeira ordem (de predicados)
   - linguagens; sintaxe; semântica (interpretações, modelos, satisfazibilidade, validade); sistema dedutivo de dedução natural; completude e integridade do *DN*; teorias e axiomatizações; indecidibilidade da lógica de primeira ordem; limite dos métodos formais (Teorema de Gödel)
3. Programação em lógica
   - resolução; fórmulas de Horn e programas definidos; unificação de termos, resolução-SLD, completude e integridade.

O que é a lógica?

A distinção entre o raciocínio correcto e incorrecto é o principal problema que aborda a lógica.

Irving Copi

A lógica estuda a razão como ferramenta do conhecimento.

Jacques Maritain

A lógica é a ciência do pensamento correcto.

Raymond McCall

O que é a lógica matemática?

$$\text{Boole} \Rightarrow \text{Frege} \Rightarrow \ldots \Rightarrow \text{Hilbert} \Rightarrow \text{Gödel} \Rightarrow \ldots$$

Formalizações simbólicas essencialmente associadas com problemas dos fundamentos da matemática...

- teoria de modelos
- teoria da dedução (ou demonstração)
- sistemas axiomáticos (p.e., teoria dos conjuntos)

Uma variedade de lógicas

- Lógica clássica: proposicional e de predicados (primeira ordem)
- Lógicas de ordem superior
- Lógicas subestruturais (p.e intuicionista): modificação das propriedades das conectivas lógicas. Ex: $\neg\neg A \not\equiv A$; $A \wedge A \not\equiv A$
- Lógicas modais: necessidade, conhecimento, temporais, descrições, etc. A noção de verdade é parametrizada... Exemplo: *Hoje é segunda-feira*
- ...

O que é "uma lógica"?

- uma linguagem de um alfabeto em que se define o que são **fórmulas** e outros objectos.
- Uma maneira de definir uma semântica (=noção do que é verdadeiro) para as fórmulas.
- Um sistema de dedução que, usando um número finito de regras permita a obtenção de fórmulas a partir de outras (=raciocínio). Deve ser íntegro: o que se deduz deve ser verdadeiro...Mas também será bom que o que é verdadeiro deve ser deduzível (completo)...Mas isto nem sempre é possível.

Lógica e Ciência de Computadores

Circuitos lógicos: lógicas Booleanas

Bases de dados: lógicas em modelos finitos (SQL=FO)

Inteligência artificial: sistemas periciais, linguagem natural, web semântica, data mining, etc. Robert Kowalski : `www.youtube.com/watch?v=H2gOQbEFtMU`

Autómatos Finitos: lógica de segunda ordem monádica com um sucessor (os modelos são conjuntos de palavras)

XML: Um documento estruturado é uma árvore de uma linguagem representada por autómatos de árvores a que correspondem fórmulas duma lógica de segunda ordem.

Algoritmos e complexidade: classes de complexidade de problemas podem ser caracterizadas por classes de fórmulas lógicas

Linguagens de programação:

- programação em lógica (p.e. `prolog`)
- teoria de tipos (sistemas tipos= sistemas dedutivos) para linguagens funcionais

Especificação formal e verificação:

- verificação de hardware (circuitos)
- testes de correção de software: lógicas de programas (dinâmicas,temporais, etc)
- cyber-sistemas
- protocolos de segurança
- sistemas multi-agentes

Demonstração automática Teorias específicas (inteiros, arrays, listas, . . .) SAT/SMT solvers

Assistentes de demonstração de teoremas Extracção de programas a partir de especificações

- Isabelle
- Coq
- KeYmaera

Lógica Proposicional

Consideram-se frases declarativas (*proposições*), que podem ser *verdadeiras* ($\top$) ou *falsas* ($\bot$)

- *Os elefantes são mamíferos* ($\top$)
- *Lisboa é uma cidade* ($\top$)
- $2 + 3 = 7$ ($\bot$)
- $5 \in \mathbb{N}$ ($\top$)
- $3 > 7$ ($\bot$)

Os elefantes são mamíferos *e* Lisboa é uma cidade $(\top)$

porque é uma conjunção de proposições $\top$

$2 + 3 = 7$ *ou* $5 \in \mathbb{N}$ $(\top)$

porque é uma disjunção de proposições das quais uma é $\top$

não $3 > 7$ $(\top)$

porque é uma negação de uma proposição $\bot$

Se $7 > 3$ *então* $3 + 3 = 6$ $(\top)$

porque uma implicação é $\top$ sse o consequente é $\top$ sempre que o antecedente é $\top$

Se $3 > 7$ *então* $2 + 3 = 6$ $(\top)$

porque é uma implicação cujo antecedente é $\bot$

Conectivas lógicas

Cada proposição vai ser representada por uma *variável proposicional* e as conectivas lógicas por *símbolos n-ários*:

Conectiva	Símbolos	Aridade	Outros símbolos equivalentes
Conjunção	$\wedge$	2	&, &&, $\cdot$
Disjunção	$\vee$	2	$\vert$, $+$
Negação	$\neg$	1	$\sim$, $^-$, !
Implicação	$\rightarrow$	2	$\Rightarrow$, $\supset$

Linguagens da lógica proposicional

Uma linguagem da lógica proposicional é formada a partir dos seguintes conjuntos de símbolos primitivos (alfabetos):

- um conjunto numerável de variáveis proposicionais $\mathcal{V}_p = \{p, q, r, \ldots, p_1, \ldots\}$
- conectivas lógicas $\wedge$, $\vee$, $\neg$, $\rightarrow$
- os parêntesis (e)

Fórmulas

Uma fórmula *bem-formada* (ϕ, ψ, θ, ...) é definida indutivamente pelas seguintes regras:

1. uma variável proposicional p é uma fórmula
2. se ϕ é uma fórmula então $(\neg\phi)$ é uma fórmula
3. se ϕ e ψ são fórmulas então $(\phi \wedge \psi)$, $(\phi \vee \psi)$ e $(\phi \rightarrow \psi)$ são fórmulas

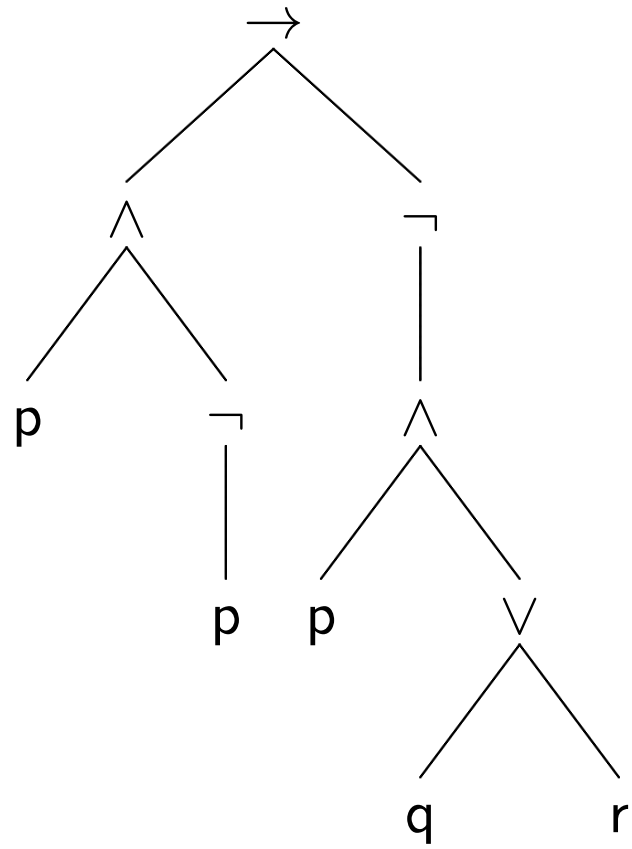
$$\phi := p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi)$$

Exemplo

$$((p \wedge (\neg p)) \rightarrow (\neg(p \wedge (q \vee r))))$$

Árvore sintáctica duma fórmula (omitindo parêntesis)

$$((p \wedge (\neg p)) \rightarrow (\neg (p \wedge (q \vee r))))$$



Convenção para omissão de parêntesis

- os parêntesis exteriores podem ser omitidos
- $\neg$ tem precedência sobre $\wedge$
- $\wedge$ tem precedência sobre $\vee$
- $\vee$ tem precedência sobre $\rightarrow$
- $\wedge$ e $\vee$ são associativas à esquerda
- $\rightarrow$ é associativa à direita

Exemplos

$\phi \wedge \psi \vee \theta$ é uma abreviatura de $((\phi \wedge \psi) \vee \theta)$
$\psi \wedge \phi \wedge \theta$ corresponde a $((\psi \wedge \phi) \wedge \theta)$
$p \wedge \neg p \rightarrow \neg(p \wedge (q \vee r))$ corresponde a $((p \wedge (\neg p)) \rightarrow (\neg(p \wedge (q \vee r))))$
$p \rightarrow q \rightarrow r$ corresponde a $p \rightarrow (q \rightarrow r)$

Sub-fórmulas

Uma sub-fórmula é definida pelas seguintes regras:

1. uma fórmula φ é sub-fórmula dela própria;
2. $\neg\phi$ tem $\neg\phi$ e todas as sub-fórmulas de ϕ como sub-fórmulas;
3. as fórmulas $\phi \wedge \psi$, $\phi \vee \psi$ e $\phi \rightarrow \psi$ têm elas próprias e todas as sub-fórmulas de ϕ e de ψ como sub-fórmulas.

Quais as sub-fórmulas de

$$\neg p \wedge q \rightarrow p \wedge (q \vee \neg r)$$

Semântica da lógica proposicional

Os valores de verdade são $\top$ e $\bot$ (em alternativa `v` e `f`, ou 1 e 0).

Atribuição de valores de verdade (ou valorização)

$v : \mathcal{V}_p \longrightarrow \{\top, \bot\}$ que atribuí um valor de verdade a cada variável

Uma valorização v pode ser estendida ao conjunto das fórmulas:

1. para $p \in \mathcal{V}_p$ o valor $v(p)$ já está definido
2. $v(\neg\phi) = \top$ sse $v(\phi) = \bot$
3. $v(\phi \wedge \psi) = \top$ sse $v(\phi) = \top$ e $v(\psi) = \top$
4. $v(\phi \vee \psi) = \top$ sse $v(\phi) = \top$ ou $v(\psi) = \top$
5. $v(\phi \rightarrow \psi) = \bot$ sse $v(\phi) = \top$ e $v(\psi) = \bot$

Tabelas de verdade

ϕ	$\neg\ \phi$
$\bot$	$\top$
$\top$	$\bot$

ϕ	ψ	$\phi \wedge\ \psi$
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

ϕ	ψ	$\phi \vee \psi$
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\top$	$\top$	$\top$

ϕ	ψ	$\phi \rightarrow \psi$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

Se $v(p) = \top$, $v(q) = \bot$ e $v(r) = \top$, podemos calcular o valor de $v((p \wedge q) \vee \neg r)$):

p	q	r	(	p	$\wedge$	q	)	$\vee$	$\neg$	r
$\top$	$\bot$	$\top$			$\bot$			$\bot$	$\bot$	

Tabela de verdade duma fórmula

A tabela de verdade de uma fórmula ϕ com n variáveis proposicionais tem 2^n linhas. Porquê? (e quantas valorizações?) Podemos construir uma tabela em que cada linha corresponde a uma delas:

p	q	r	(p $\wedge$ q) $\vee$ $\neg$ r
$\top$	$\top$	$\top$	$\top$ $\top$ $\bot$
$\top$	$\top$	$\bot$	$\top$ $\top$ $\top$
$\top$	$\bot$	$\top$	$\bot$ $\bot$ $\bot$
$\top$	$\bot$	$\bot$	$\bot$ $\top$ $\top$
$\bot$	$\top$	$\top$	$\bot$ $\bot$ $\bot$
$\bot$	$\top$	$\bot$	$\bot$ $\top$ $\top$
$\bot$	$\bot$	$\top$	$\bot$ $\bot$ $\bot$
$\bot$	$\bot$	$\bot$	$\bot$ $\top$ $\top$

Relação de satisfazibilidade

Seja v uma valorização, a relação $\models_v$ pode ser definida indutivamente na estrutura de ϕ por:

1. $\models_v p$ sse $v(p) = \top$;
2. $\models_v \neg\phi$ sse $\not\models_v \phi$;
3. $\models_v \phi \wedge \psi$ sse $\models_v \phi$ e $\models_v \psi$;
4. $\models_v \phi \vee \psi$ sse $\models_v \phi$ ou $\models_v \psi$;
5. $\models_v \phi \rightarrow \psi$ sse $\not\models_v \phi$ ou $\models_v \psi$;

Dizemos que v *satisfaz* ϕ sse $\models_v \phi$.

Exemplo

Sendo $v(p) = \top$, $v(q) = v(r) = \bot$ determine se

$$\models_v (p \rightarrow (q \vee r)) \vee (r \rightarrow \neg p).$$

Para tal, $\models_v p \rightarrow (q \vee r)$ ou $\models_v r \rightarrow \neg p$.
A segunda verifica-se porque $\not\models_v r$.
Pelo que podemos também concluir que

$$\models_v (p \rightarrow (q \vee r)) \vee (r \rightarrow \neg p).$$

Exemplo

Determinar se $(p \wedge q) \vee \neg r$ é válida, isto é, $\models (p \wedge q) \vee \neg r$.
Mas,
se $v(r) = \top$ e $v(p) = \bot$ então $\not\models_v (p \wedge q) \vee \neg r$: porque $\not\models_v \neg r$ e $\not\models_v (p \wedge q)$.
Logo, não é válida.

Satisfazibilidade, Tautologias e Contradições

Uma fórmula ϕ é

satisfazível se existe uma valorização v tal que $v(\phi) = \top$,
escreve-se $\models_v \phi$ e diz-se que v satisfaz ϕ

tautologia se para todas as valorizações v, $v(\phi) = \top$ e
escreve-se $\models \phi$ Ex: $\models p \vee \neg p$ (Terceiro excluído)

contradição se para todas as valorizações v, $v(\phi) = \bot$
Ex: $p \wedge \neg p$ é uma contradição.

Escrevemos $\not\models \phi$ sse ϕ não é uma tautologia.
Uma tautologia chama-se também uma fórmula válida.
Uma fórmula é não-satisfazível se e só se é uma contradição.

Relacione as afirmações seguintes:

$$\not\models \varphi$$

e

$$\models \neg\varphi$$

Consequência e equivalência semântica

Satisfazibilidade de um conjunto de fórmulas

Seja Γ um conjunto de fórmulas.
Uma valorização v satisfaz Γ se e só se v satisfaz toda a fórmula $\psi \in \Gamma$.

$$\forall \psi \in \Gamma,\ \models_v \psi$$

Γ é satisfazível se existe uma valorização que o satisfaz

Consequência

Uma fórmula ϕ é uma consequência semântica de Γ se para toda a valorização v que satisfaz Γ, se tem $v(\phi) = \top$; e escreve-se $\Gamma \models \phi$

Exercício:

As afirmações seguintes são verdadeiras ou falsas? Justifique!!!

1) $p \vee q, \neg q \vee r \models p \vee r$

2) $p \vee q, \neg q \vee r \models p \wedge r$

$\emptyset \models \phi$ é equivalente a $\models \phi$

Se $\Gamma = \{\psi\}$ e $\Gamma \models \phi$ então diz-se que ϕ é consequência semântica de ψ ($\psi \models \phi$)

Fórmulas semânticamente equivalentes

Equivalência semântica

Se $\psi \models \phi$ e $\phi \models \psi$ então ψ e ϕ são semânticamente equivalentes e escreve-se $\psi \equiv \phi$.

$\phi \wedge \psi$	$\equiv$	$\psi \wedge \phi$	(comutatividade)
$\phi \vee \psi$	$\equiv$	$\psi \vee \phi$	(comutatividade)
$\neg(\phi \wedge \psi)$	$\equiv$	$(\neg\phi \vee \neg\psi)$	(Lei de DeMorgan)
$\neg(\phi \vee \psi)$	$\equiv$	$(\neg\phi \wedge \neg\psi)$	(Lei de DeMorgan)
$(\phi \wedge \psi) \wedge \theta$	$\equiv$	$\phi \wedge (\psi \wedge \theta)$	(associatividade)
$(\phi \vee \psi) \vee \theta$	$\equiv$	$\phi \vee (\psi \vee \theta)$	(associatividade)
$(\phi \vee \phi)$	$\equiv$	ϕ	(idempotência)
$(\phi \wedge \phi)$	$\equiv$	ϕ	(idempotência)
$(\phi \wedge \psi) \vee \theta$	$\equiv$	$(\phi \vee \theta) \wedge (\psi \vee \theta)$	(distributividade)
$(\phi \vee \psi) \wedge \theta$	$\equiv$	$(\phi \wedge \theta) \vee (\psi \wedge \theta)$	(distributividade)
$\neg\neg\phi$	$\equiv$	ϕ	(Dupla negação)
$\phi \rightarrow \psi$	$\equiv$	$\neg\phi \vee \psi$	