Logique

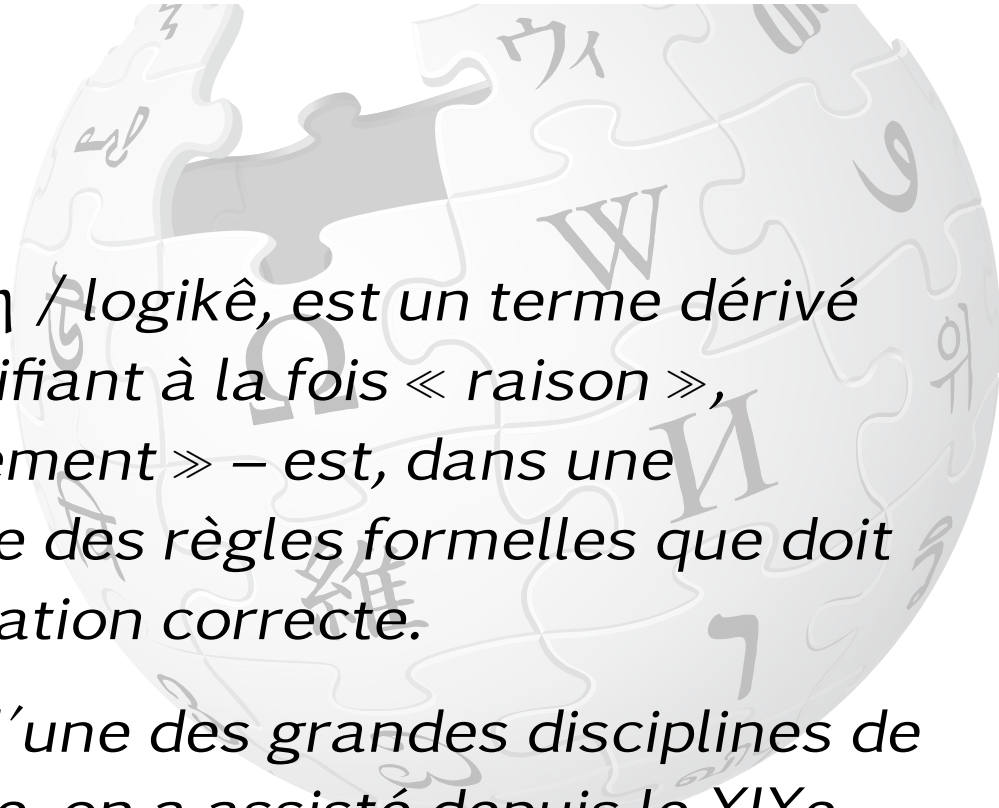
1. Introduction & Syntaxe

Sylvain Schmitz

Licence 3e année
UFR Informatique

12 septembre 2019

Wikipedia



La logique, du grec λογικη / logikê, est un terme dérivé de λογολσ / lógos — signifiant à la fois « raison », « langage » et « raisonnement » – est, dans une première approche, l'étude des règles formelles que doit respecter toute argumentation correcte.

Elle est depuis l'Antiquité l'une des grandes disciplines de la philosophie [...]. En outre, on a assisté depuis le XIXe siècle au développement fulgurant d'une approche mathématique de la logique. Sa convergence opérée avec l'informatique depuis la fin du XXe siècle lui a donné un regain de vitalité.

Logique philosophique

quels sont les raisonnements concluants ?

- « Tous les humains sont mortels, or Socrate est humain, donc Socrate est mortel »
- « Toutes les souris sont vertes, or Yoda est une souris, donc Yoda est vert »
- « Tous les humains sont mortels, or Socrate est mortel, donc Socrate est humain »

Logique philosophique

quels sont les raisonnements concluants ?

- « Tous les humains sont mortels, or Socrate est humain, donc Socrate est mortel »

formule logique valide :

$$\big((\forall x\,.\,H(x) \Rightarrow M(x)) \wedge H(s)\big) \Rightarrow M(s)$$

Logique philosophique

quels sont les raisonnements concluants ?

- « Tous les humains sont mortels, or Socrate est humain, donc Socrate est mortel »

formule logique valide :

$$\big((\forall x\,.\,H(x) \Rightarrow M(x)) \wedge H(s)\big) \Rightarrow M(s)$$

Logique mathématique

- depuis le XIXe siècle : formalisation des mathématiques
- « crise des fondements »

  Paradoxes

  Un barbier rase tous les hommes qui ne se rasent pas eux-mêmes (et seulement ceux-là) ; est-ce que ce barbier se rase lui-même ?

- développement de la logique

Logique informatique

Par ma foi ! Il y a plus de quarante ans que je dis de la prose sans que j'en susse rien.

Le bourgeois gentilhomme, Molière

Expressions booléennes

formule logique	expression Java
$(P \wedge \neg Q) \vee P$	`(P && !Q) || P`
$\neg(x = 0) \wedge x \cdot y = 0$	`!(x == 0) && x*y = 0`

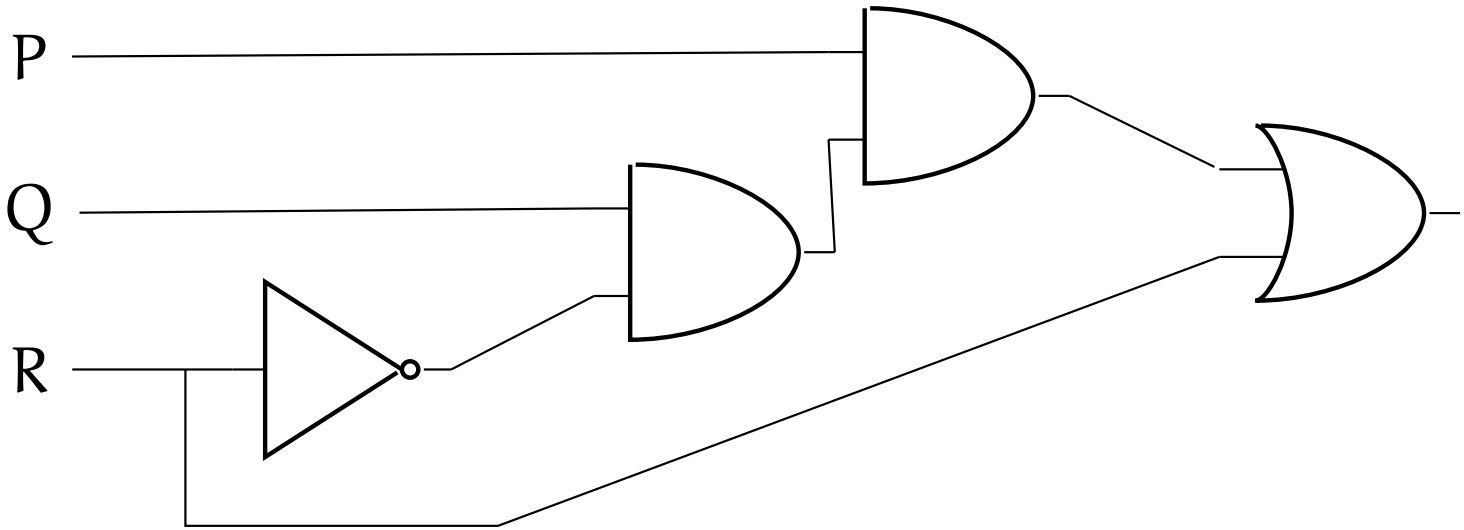
- peut-on simplifier ces expressions ?

Expressions booléennes

formule logique	expression Java
$(P \wedge \neg Q) \vee P$	`(P && !Q) || P`
$\neg(x = 0) \wedge x \cdot y = 0$	`!(x == 0) && x*y = 0`

- peut-on simplifier ces expressions ?

Circuits booléens

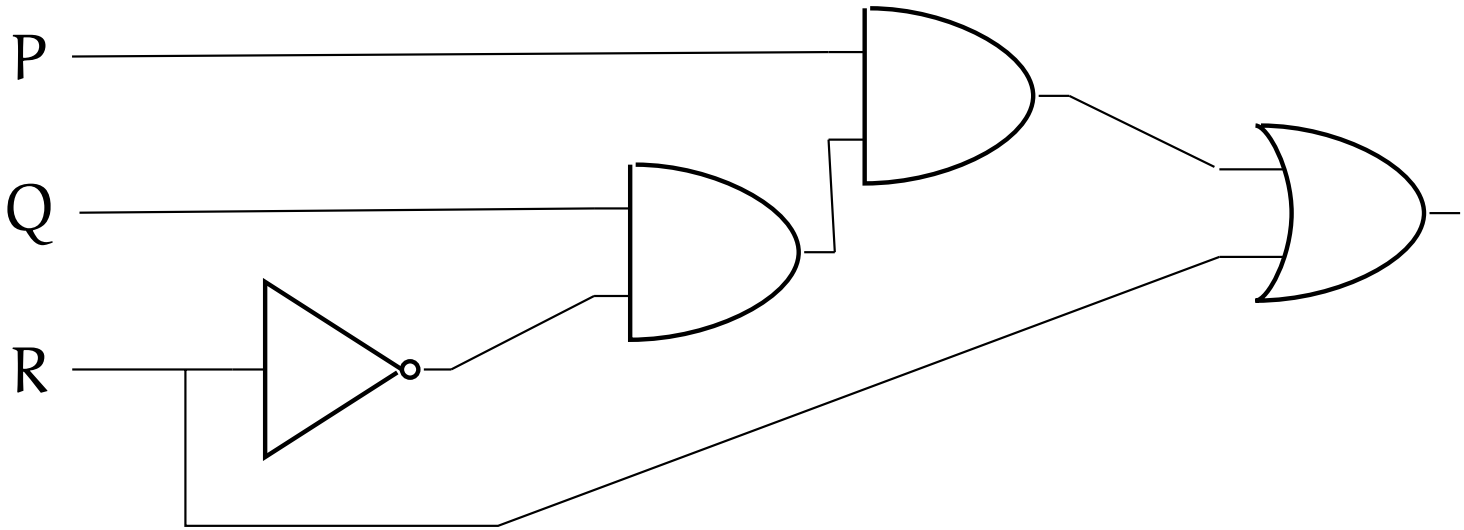


réalisation matérielle de la formule propositionnelle

$$(P \wedge (Q \wedge \neg R)) \vee R$$

- comment construire un circuit ?
- peut-on minimiser un circuit ?
- cours « circuits et architecture » en M1

Circuits booléens

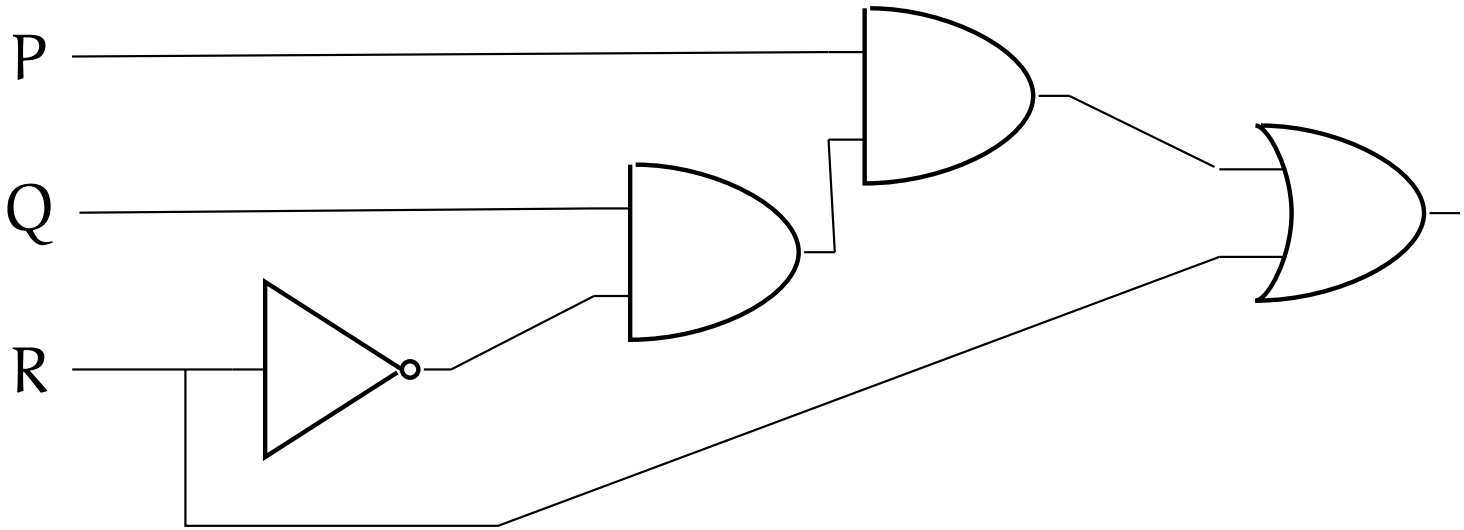


réalisation matérielle de la formule propositionnelle

$$(P \wedge (Q \wedge \neg R)) \vee R$$

- comment construire un circuit ?
- peut-on minimiser un circuit ?
- cours « circuits et architecture » en M1

Circuits booléens



réalisation matérielle de la formule propositionnelle

$$(P \wedge (Q \wedge \neg R)) \vee R$$

- comment construire un circuit ?
- peut-on minimiser un circuit ?
- cours « circuits et architecture » en M1

Programmer ≈ prouver

L'isomorphisme de Curry-Howard au cœur du fonctionnement des assistants de preuve et de la théorie des types.

- cours « preuves assistées par ordinateur » en M1

Programmer ≈ prouver

L'isomorphisme de Curry-Howard au cœur du fonctionnement des assistants de preuve et de la théorie des types.

- cours ≪ preuves assistées par ordinateur ≫ en M1

Complexité algorithmique

problèmes logiques emblématiques des problèmes rencontrés en informatique

problème logique	complexité
2SAT	NL-complet
HornSAT	P-complet
SAT	NP-complet
QBF	PSPACE-complet

- les problèmes informatiques s'encodent comme des problèmes logiques
- cours « calculabilité et complexité » en M1

Complexité algorithmique

problèmes logiques emblématiques des problèmes rencontrés en informatique

problème logique	complexité
2SAT	NL-complet
HornSAT	P-complet
SAT	NP-complet
QBF	PSPACE-complet

- les problèmes informatiques s'encodent comme des problèmes logiques
- cours « calculabilité et complexité » en M1

Complexité algorithmique

problèmes logiques emblématiques des problèmes rencontrés en informatique

problème logique	complexité
2SAT	NL-complet
HornSAT	P-complet
SAT	NP-complet
QBF	PSPACE-complet

- les problèmes informatiques s'encodent comme des problèmes logiques
- cours « calculabilité et complexité » en M1

La révolution SAT

- solveurs pour la satisfiabilité de formules propositionnelles
- extrêmement performants

Résolution de contraintes

Exemple : sudoku

démo : `https://blag.cedeela.fr/sudoku-msat/`

					3		8	5
		1		2				
			5		7			
		4				1		
	9							
5							7	3
		2		1				
				4				9

- aussi : faire un emploi du temps, planifier des tâches, trouver une installation optimale de paquets linux, etc.
- cours « programmation logique et par contraintes » en M1

Résolution de contraintes

Exemple : sudoku

démo : `https://blag.cedeela.fr/sudoku-msat/`

9	8	7	6	5	4	3	2	1
2	4	6	1	7	3	9	8	5
3	5	1	9	2	8	7	4	6
1	2	8	5	3	7	6	9	4
6	3	4	8	9	2	1	5	7
7	9	5	4	6	1	8	3	2
5	1	9	2	8	6	4	7	3
4	7	2	3	1	9	5	6	8
8	6	3	7	4	5	2	1	9

- aussi : faire un emploi du temps, planifier des tâches, trouver une installation optimale de paquets linux, etc.
- cours « programmation logique et par contraintes » en M1

Résolution de contraintes

Exemple : sudoku

démo : `https://blag.cedeela.fr/sudoku-msat/`

9	8	7	6	5	4	3	2	1
2	4	6	1	7	3	9	8	5
3	5	1	9	2	8	7	4	6
1	2	8	5	3	7	6	9	4
6	3	4	8	9	2	1	5	7
7	9	5	4	6	1	8	3	2
5	1	9	2	8	6	4	7	3
4	7	2	3	1	9	5	6	8
8	6	3	7	4	5	2	1	9

- aussi : faire un emploi du temps, planifier des tâches, trouver une installation optimale de paquets linux, etc.
- cours « programmation logique et par contraintes » en M1

Résolution de contraintes

Exemple : sudoku

démo : `https://blag.cedeela.fr/sudoku-msat/`

9	8	7	6	5	4	3	2	1
2	4	6	1	7	3	9	8	5
3	5	1	9	2	8	7	4	6
1	2	8	5	3	7	6	9	4
6	3	4	8	9	2	1	5	7
7	9	5	4	6	1	8	3	2
5	1	9	2	8	6	4	7	3
4	7	2	3	1	9	5	6	8
8	6	3	7	4	5	2	1	9

- aussi : faire un emploi du temps, planifier des tâches, trouver une installation optimale de paquets linux, etc.
- cours ≪ programmation logique et par contraintes ≫ en M1

Base de données

- une requête SQL

  SELECT Vols.depart
  FROM Vols **JOIN** Aeroports **ON** Vols.arrivee = Aeroports.nom
  WHERE Aeroports.pays = 'FR'

- . . . n'est jamais qu'une formule logique

$$q(d) = \exists a \exists n \exists p \,.\, V(d,a) \wedge A(n,p) \wedge a = n \wedge p = \text{'FR'}$$

- comment évaluer une requête ?
- peut-on optimiser les requêtes ?
- cours de « bases de données avancées »

Base de données

- une requête SQL

  SELECT Vols.depart
  FROM Vols **JOIN** Aeroports **ON** Vols.arrivee = Aeroports.nom
  WHERE Aeroports.pays = 'FR'

- . . . n'est jamais qu'une formule logique

$$q(d) = \exists a \exists n \exists p \,.\, V(d,a) \wedge A(n,p) \wedge a = n \wedge p = \text{'FR'}$$

- comment évaluer une requête ?
- peut-on optimiser les requêtes ?
- cours de « bases de données avancées »

Base de données

- une requête SQL

  SELECT Vols.depart
  FROM Vols **JOIN** Aeroports **ON** Vols.arrivee = Aeroports.nom
  WHERE Aeroports.pays = 'FR'

- . . . n'est jamais qu'une formule logique

$$q(d) = \exists a \exists n \exists p \,.\, V(d,a) \wedge A(n,p) \wedge a = n \wedge p = \text{'FR'}$$

- comment évaluer une requête ?
- peut-on optimiser les requêtes ?
- cours de « bases de données avancées »

Base de données

- une requête SQL

  SELECT Vols.depart
  FROM Vols **JOIN** Aeroports **ON** Vols.arrivee = Aeroports.nom
  WHERE Aeroports.pays = 'FR'

- . . . n'est jamais qu'une formule logique

$$q(d) = \exists a \exists n \exists p \,.\, V(d,a) \wedge A(n,p) \wedge a = n \wedge p = \text{'FR'}$$

- comment évaluer une requête ?
- peut-on optimiser les requêtes ?
- cours de « bases de données avancées »

Organisation

- la page du cours

  `https://www.irif.fr/~schmitz/teach/2019_lo5/`

  contient les notes de cours, les transparents, les feuilles de TD, les dates d'examen et les consignes

- les intervenants :
  - Antonio Bucciarelli (buccia@irif.fr)
  - Rémi Nollet (nollet@irif.fr)
  - Daniela Petrisan (petrisan@irif.fr)
  - Sylvain Schmitz (schmitz@irif.fr)
  - Sam van Gool (samvangool@irif.fr)

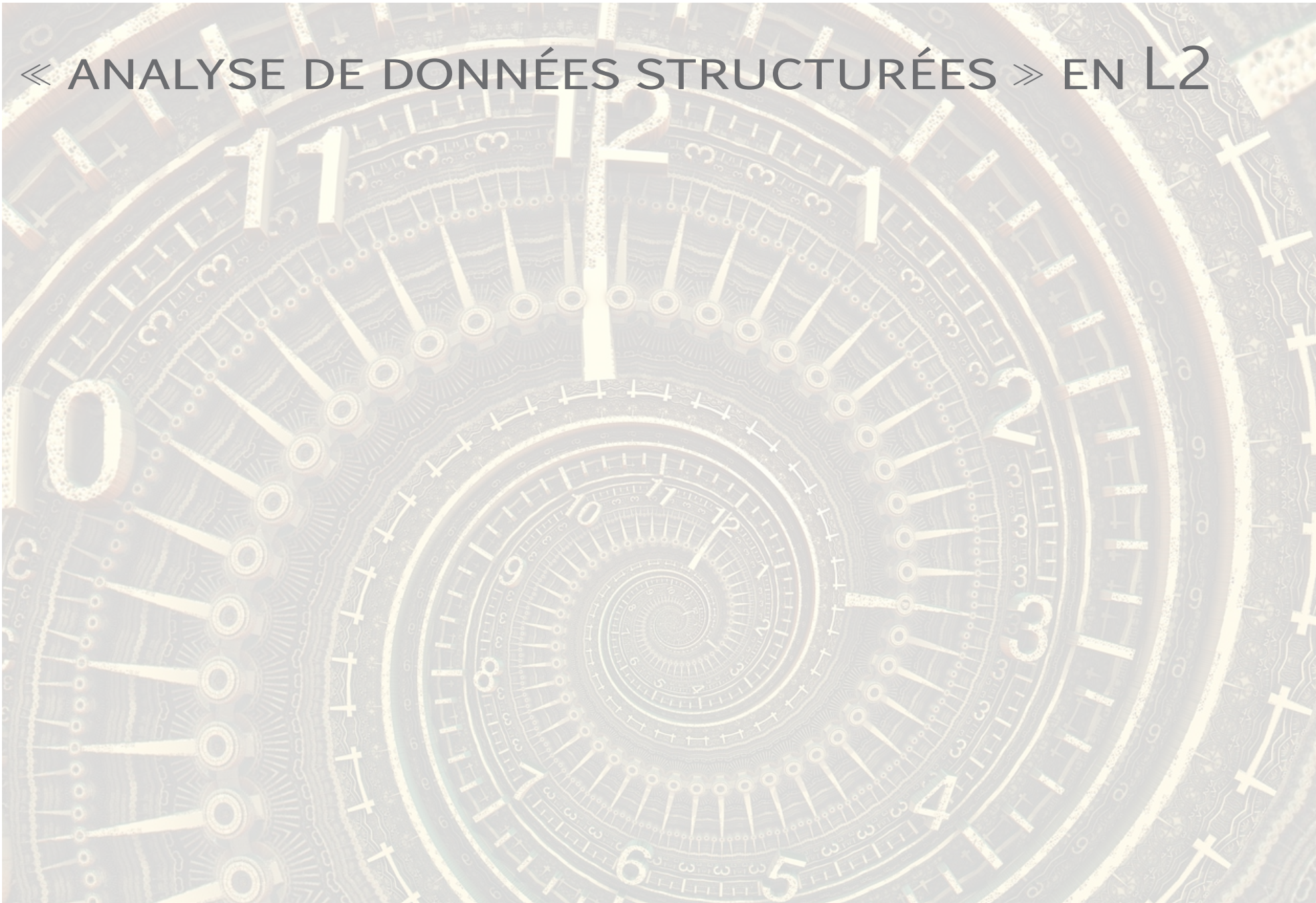
Modalités de contrôle des connaissances

- contrôle continu
  - $\mathrm{QCM}_1, \mathrm{QCM}_2$: deux QCM
  - $\mathrm{mP}_1, \mathrm{mP}_2, \mathrm{mP}_3$: trois rendus de mini-projets en binômes (même binôme pour les trois rendus)
- E : examen

  note de première session :

$$\frac{1}{10}\mathrm{QCM}_1 + \frac{1}{10}\mathrm{mP}_1 + \frac{1}{10}\mathrm{mP}_2 + \frac{1}{10}\mathrm{QCM}_2 + \frac{1}{10}\mathrm{mP}_3 + \frac{5}{10}\mathrm{E}$$

« Analyse de données structurées » en L2



« Analyse de données structurées » en L2

Syntaxe

expression E

Sémantique

valeur $[\![E]\!]^C$
dans un contexte C

« Analyse de données structurées » en L2

Syntaxe

expression E

syntaxe concrète

syntaxe abstraite

Sémantique

valeur $[\![E]\!]^C$

dans un contexte C

« Analyse de données structurées » en L2

Syntaxe

expression E

syntaxe concrète

$1 + y \times 3$

syntaxe abstraite

Sémantique

valeur $[\![E]\!]^C$

dans un contexte C

« Analyse de données structurées » en L2

Syntaxe

expression E

syntaxe concrète

$1 + y \times 3$

syntaxe abstraite

Sémantique

valeur $[\![E]\!]^C$

dans un contexte C

« Analyse de données structurées » en L2

Syntaxe

expression E

syntaxe concrète

$1 + y \times 3$

syntaxe abstraite

Sémantique

valeur $[\![E]\!]^C$
dans un contexte C

$[\![E]\!]^{[4/y]} =$

« Analyse de données structurées » en L2

Syntaxe

expression E

syntaxe concrète

$1 + y \times 3$

syntaxe abstraite

Sémantique

valeur $[\![E]\!]^C$

dans un contexte C

$[\![E]\!]^{[4/y]} = 13$

Logique propositionnelle

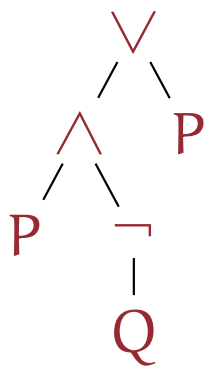
Syntaxe

formule φ

syntaxe concrète

$(P \wedge \neg Q) \vee P$

syntaxe abstraite



Sémantique

valeur $[\![\varphi]\!]^C$
dans un contexte C

$[\![\varphi]\!]^{[\bot/P,\bot/Q]} = \bot$

Syntaxe

Soit $\mathcal{P}_0$ un ensemble dénombrable de propositions.
Les formules propositionnelles sont définies par la syntaxe abstraite

$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

où $P \in \mathcal{P}_0$.

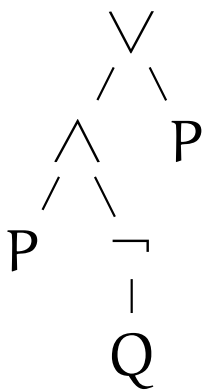
Syntaxe

Soit $\mathcal{P}_0$ un ensemble dénombrable de propositions.
Les formules propositionnelles sont définies par la syntaxe abstraite

$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

où $P \in \mathcal{P}_0$.

une formule



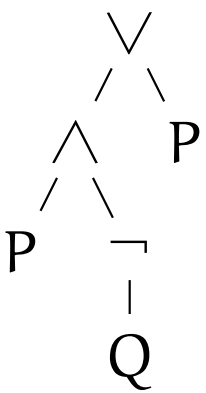
Syntaxe

Soit $\mathcal{P}_0$ un ensemble dénombrable de propositions.
Les formules propositionnelles sont définies par la syntaxe abstraite

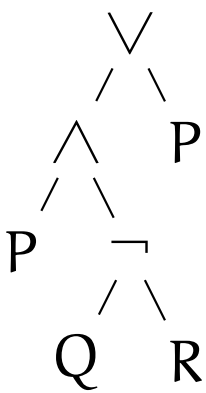
$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

où $P \in \mathcal{P}_0$.

une formule



pas une formule



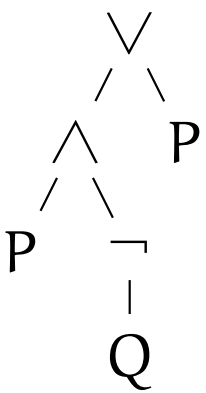
Syntaxe

Soit $\mathcal{P}_0$ un ensemble dénombrable de propositions.
Les formules propositionnelles sont définies par la syntaxe abstraite

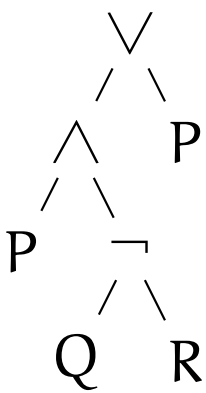
$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

où $P \in \mathcal{P}_0$.

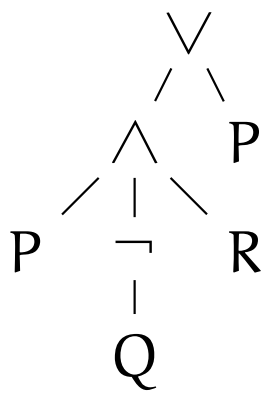
une formule



pas une formule



pas une formule



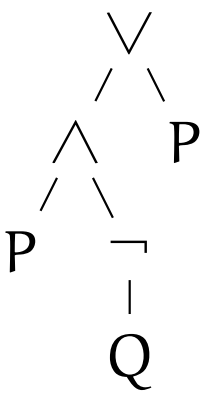
Syntaxe

Soit $\mathcal{P}_0$ un ensemble dénombrable de propositions.
Les formules propositionnelles sont définies par la syntaxe abstraite

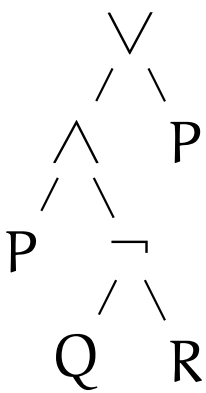
$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

où $P \in \mathcal{P}_0$.

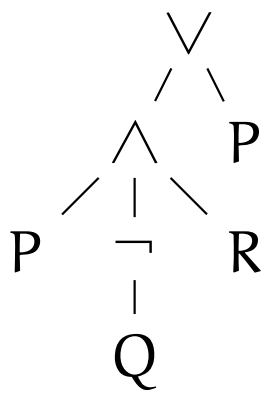
une formule



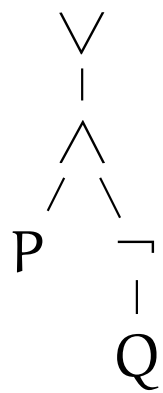
pas une formule



pas une formule



pas une formule



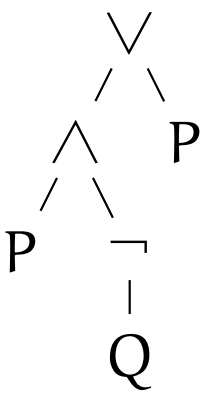
Syntaxe

Soit $\mathcal{P}_0$ un ensemble dénombrable de propositions.
Les formules propositionnelles sont définies par la syntaxe abstraite

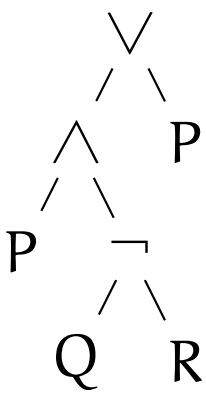
$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

où $P \in \mathcal{P}_0$.

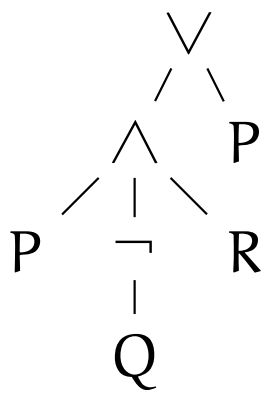
une formule



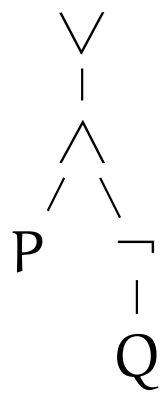
pas une formule



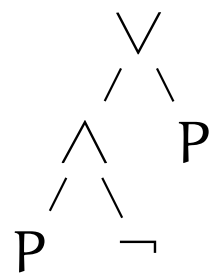
pas une formule



pas une formule



pas une formule



Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

Par induction structurelle sur φ.

- cas de base : si $\varphi = P \in \mathcal{P}_0$ alors
$$\mathit{feuilles}(\varphi) = 1 \leqslant 0 + 1 = 2 \cdot \mathit{internes}(\varphi) + 1$$
- étape d'induction : si $\varphi = \neg\psi$ alors
$$\begin{aligned} \mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi) \\ &\leqslant 2 \cdot \mathit{internes}(\psi) + 1 && \text{(par hyp. ind.)} \\ &\leqslant 2 \cdot (\mathit{internes}(\psi) + 1) + 1 \\ &= 2 \cdot \mathit{internes}(\varphi) + 1 \end{aligned}$$

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

Par induction structurelle sur φ.

- cas de base : si $\varphi = P \in \mathcal{P}_0$ alors
$$\mathit{feuilles}(\varphi) = 1 \leqslant 0 + 1 = 2 \cdot \mathit{internes}(\varphi) + 1$$
- étape d'induction : si $\varphi = \neg\psi$ alors
$$\begin{aligned} \mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi) \\ &\leqslant 2 \cdot \mathit{internes}(\psi) + 1 && \text{(par hyp. ind.)} \\ &\leqslant 2 \cdot (\mathit{internes}(\psi) + 1) + 1 \\ &= 2 \cdot \mathit{internes}(\varphi) + 1 \end{aligned}$$

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

Par induction structurelle sur φ.

- cas de base : si $\varphi = P \in \mathcal{P}_0$ alors
$$\mathit{feuilles}(\varphi) = 1 \leqslant 0 + 1 = 2 \cdot \mathit{internes}(\varphi) + 1$$
- étape d'induction : si $\varphi = \neg\psi$ alors
$$\begin{aligned} \mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi) \\ &\leqslant 2 \cdot \mathit{internes}(\psi) + 1 && \text{(par hyp. ind.)} \\ &\leqslant 2 \cdot (\mathit{internes}(\psi) + 1) + 1 \\ &= 2 \cdot \mathit{internes}(\varphi) + 1 \end{aligned}$$

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

Par induction structurelle sur φ.

- cas de base : si $\varphi = P \in \mathcal{P}_0$ alors
$$\mathit{feuilles}(\varphi) = 1 \leqslant 0 + 1 = 2 \cdot \mathit{internes}(\varphi) + 1$$
- étape d'induction : si $\varphi = \neg\psi$ alors
$$\begin{aligned} \mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi) \\ &\leqslant 2 \cdot \mathit{internes}(\psi) + 1 && \text{(par hyp. ind.)} \\ &\leqslant 2 \cdot (\mathit{internes}(\psi) + 1) + 1 \\ &= 2 \cdot \mathit{internes}(\varphi) + 1 \end{aligned}$$

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

Par induction structurelle sur φ.

- cas de base : si $\varphi = P \in \mathcal{P}_0$ alors
$$\mathit{feuilles}(\varphi) = 1 \leqslant 0 + 1 = 2 \cdot \mathit{internes}(\varphi) + 1$$
- étape d'induction : si $\varphi = \neg\psi$ alors
$$\begin{aligned} \mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi) \\ &\leqslant 2 \cdot \mathit{internes}(\psi) + 1 && \text{(par hyp. ind.)} \\ &\leqslant 2 \cdot (\mathit{internes}(\psi) + 1) + 1 \\ &= 2 \cdot \mathit{internes}(\varphi) + 1 \end{aligned}$$

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

Par induction structurelle sur φ.

- cas de base : si $\varphi = P \in \mathcal{P}_0$ alors
$$\mathit{feuilles}(\varphi) = 1 \leqslant 0 + 1 = 2 \cdot \mathit{internes}(\varphi) + 1$$
- étape d'induction : si $\varphi = \neg\psi$ alors
$$\begin{aligned} \mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi) \\ &\leqslant 2 \cdot \mathit{internes}(\psi) + 1 && \text{(par hyp. ind.)} \\ &\leqslant 2 \cdot (\mathit{internes}(\psi) + 1) + 1 \\ &= 2 \cdot \mathit{internes}(\varphi) + 1 \end{aligned}$$

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

Par induction structurelle sur φ.

- cas de base : si $\varphi = P \in \mathcal{P}_0$ alors
$$\mathit{feuilles}(\varphi) = 1 \leqslant 0 + 1 = 2 \cdot \mathit{internes}(\varphi) + 1$$
- étape d'induction : si $\varphi = \neg\psi$ alors
$$\begin{aligned} \mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi) \\ &\leqslant 2 \cdot \mathit{internes}(\psi) + 1 && \text{(par hyp. ind.)} \\ &\leqslant 2 \cdot (\mathit{internes}(\psi) + 1) + 1 \\ &= 2 \cdot \mathit{internes}(\varphi) + 1 \end{aligned}$$

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

Par induction structurelle sur φ.

- cas de base : si $\varphi = P \in \mathcal{P}_0$ alors
$$\mathit{feuilles}(\varphi) = 1 \leqslant 0 + 1 = 2 \cdot \mathit{internes}(\varphi) + 1$$
- étape d'induction : si $\varphi = \neg\psi$ alors
$$\begin{aligned} \mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi) \\ &\leqslant 2 \cdot \mathit{internes}(\psi) + 1 && \text{(par hyp. ind.)} \\ &\leqslant 2 \cdot (\mathit{internes}(\psi) + 1) + 1 \\ &= 2 \cdot \mathit{internes}(\varphi) + 1 \end{aligned}$$

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

$\vdots$

- étape d'induction : si $\varphi = \psi_1 \vee \psi_2$ alors

$$\begin{aligned}
\mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi_1) + \mathit{feuilles}(\psi_2) \\
&\leqslant 2 \cdot \mathit{internes}(\psi_1) + 1 + 2 \cdot \mathit{internes}(\psi_2) + 1 \quad \text{(par hyp. ind.)} \\
&= 2 \cdot (\mathit{internes}(\psi_1) + \mathit{internes}(\psi_2) + 1) \\
&= 2 \cdot \mathit{internes}(\varphi) \\
&\leqslant 2 \cdot \mathit{internes}(\varphi) + 1
\end{aligned}$$

- étape d'induction : si $\varphi = \psi_1 \wedge \psi_2$ similaire □

Raisonnement inductif : exemple

Proposition

Pour toute formule propositionnelle φ, les nombres de feuilles et de nœuds internes sont reliés par l'inégalité :

$$\mathit{feuilles}(\varphi) \leqslant 2 \cdot \mathit{internes}(\varphi) + 1$$

Démonstration.

$\vdots$

- étape d'induction : si $\varphi = \psi_1 \vee \psi_2$ alors

$$\begin{aligned}
\mathit{feuilles}(\varphi) &= \mathit{feuilles}(\psi_1) + \mathit{feuilles}(\psi_2) \\
&\leqslant 2 \cdot \mathit{internes}(\psi_1) + 1 + 2 \cdot \mathit{internes}(\psi_2) + 1 \quad \text{(par hyp. ind.)} \\
&= 2 \cdot (\mathit{internes}(\psi_1) + \mathit{internes}(\psi_2) + 1) \\
&= 2 \cdot \mathit{internes}(\varphi) \\
&\leqslant 2 \cdot \mathit{internes}(\varphi) + 1
\end{aligned}$$

- étape d'induction : si $\varphi = \psi_1 \wedge \psi_2$ similaire □

Définition inductive : exemple

$\mathcal{P}_0(\varphi) \subseteq \mathcal{P}_0$ est l'ensemble des propositions qui apparaissent au moins une fois dans une formule φ :

$$\mathcal{P}_0(P) \stackrel{\text{def}}{=} \{P\}, \qquad \mathcal{P}_0(\neg\varphi) \stackrel{\text{def}}{=} \mathcal{P}_0(\varphi),$$

$$\mathcal{P}_0(\varphi \vee \psi) \stackrel{\text{def}}{=} \mathcal{P}_0(\varphi) \cup \mathcal{P}_0(\psi) \qquad \mathcal{P}_0(\varphi \wedge \psi) \stackrel{\text{def}}{=} \mathcal{P}_0(\varphi) \cup \mathcal{P}_0(\psi).$$

Représentation en Java

```
public abstract class Formule {
    // sous-classes
    public static class Et extends Formule {
        private Formule phi1; // sous-formule gauche
        private Formule phi2; // sous-formule droite
        public Et (Formule phi1, Formule phi2) {
            this.phi1 = phi1;
            this.phi2 = phi2;
        }
        // methodes pour formules ∧
        ⋮
    }
    public static class Ou extends Formule {
        private Formule phi1; // sous-formule gauche
        private Formule phi2; // sous-formule droite
        public Ou (Formule phi1, Formule phi2) {
            this.phi1 = phi1;
            this.phi2 = phi2;
        }
        // methodes pour formules ∨
        ⋮
    }
    public static class Non extends Formule {
        private Formule phi1; // sous-formule
        public Non (Formule phi1) {
            this.phi1 = phi1;
        }
        // methodes pour formules ¬
        ⋮
    }
    public static class Proposition extends Formule {
        private String nom; // nom de la proposition
        public Proposition (String nom) {
            this.nom = nom;
        }
        // methodes pour propositions
        ⋮
    }
    // methodes de la classe abstraite
    ⋮
}
```

Méthode récursive en Java : $\mathcal{P}_0(\varphi)$

- dans Formule :

```
public abstract Set<String> getPropositions();
```

- dans Formule.Proposition :

```
public Set<String> getPropositions() {
    Set<String> propositions = new TreeSet<String>();
    propositions.add(nom);
    return propositions;
}
```

- dans Formule.Non :

```
public Set<String> getPropositions() {
    return phi1.getPropositions();
}
```

- dans Formule.Et et Formule.Ou :

```
public Set<String> getPropositions() {
    Set<String> propositions = phi1.getPropositions();
    propositions.addAll(phi2.getPropositions());
    return propositions;
}
```

Méthode récursive en Java : $\mathcal{P}_0(\varphi)$

- dans Formule :

```
public abstract Set<String> getPropositions();
```

- dans Formule.Proposition :

```
public Set<String> getPropositions() {
    Set<String> propositions = new TreeSet<String>();
    propositions.add(nom);
    return propositions;
}
```

- dans Formule.Non :

```
public Set<String> getPropositions() {
    return phi1.getPropositions();
}
```

- dans Formule.Et et Formule.Ou :

```
public Set<String> getPropositions() {
    Set<String> propositions = phi1.getPropositions();
    propositions.addAll(phi2.getPropositions());
    return propositions;
}
```

Méthode récursive en Java : $\mathcal{P}_0(\varphi)$

- dans Formule :

```
public abstract Set<String> getPropositions();
```

- dans Formule.Proposition :

```
public Set<String> getPropositions() {
    Set<String> propositions = new TreeSet<String>();
    propositions.add(nom);
    return propositions;
}
```

- dans Formule.Non :

```
public Set<String> getPropositions() {
    return phi1.getPropositions();
}
```

- dans Formule.Et et Formule.Ou :

```
public Set<String> getPropositions() {
    Set<String> propositions = phi1.getPropositions();
    propositions.addAll(phi2.getPropositions());
    return propositions;
}
```

Méthode récursive en Java : $\mathcal{P}_0(\varphi)$

- dans Formule :

```
public abstract Set<String> getPropositions();
```

- dans Formule.Proposition :

```
public Set<String> getPropositions() {
    Set<String> propositions = new TreeSet<String>();
    propositions.add(nom);
    return propositions;
}
```

- dans Formule.Non :

```
public Set<String> getPropositions() {
    return phi1.getPropositions();
}
```

- dans Formule.Et et Formule.Ou :

```
public Set<String> getPropositions() {
    Set<String> propositions = phi1.getPropositions();
    propositions.addAll(phi2.getPropositions());
    return propositions;
}
```

Syntaxe concrète

- parenthèses $\approx$ structure de l'arbre de syntaxe abstraite
- exemple : $((P \wedge \neg Q) \vee P)$
- exemples :
  - $(P \wedge \neg Q) \vee P$
  - $P \wedge \neg Q \wedge R$
  - $\neg P \vee Q \vee R$
- non-exemples :
  - $P(\vee Q$
  - $P \neg Q$
  - $P \wedge \neg Q \vee R$

Syntaxe concrète

- parenthèses $\approx$ structure de l'arbre de syntaxe abstraite
- exemple : $((P \wedge \neg Q) \vee P)$
- exemples :
  - $(P \wedge \neg Q) \vee P$
  - $P \wedge \neg Q \wedge R$
  - $\neg P \vee Q \vee R$
- non-exemples :
  - $P(\vee Q$
  - $P\neg Q$
  - $P \wedge \neg Q \vee R$

Syntaxe concrète

- parenthèses $\approx$ structure de l'arbre de syntaxe abstraite
- exemple : $((P \wedge \neg Q) \vee P)$
- exemples :
  - $(P \wedge \neg Q) \vee P$
  - $P \wedge \neg Q \wedge R$
  - $\neg P \vee Q \vee R$
- non-exemples :
  - $P(\vee Q$
  - $P\neg Q$
  - $P \wedge \neg Q \vee R$

Syntaxe concrète

- parenthèses $\approx$ structure de l'arbre de syntaxe abstraite
- exemple : $((P \wedge \neg Q) \vee P)$
- exemples :
  - $(P \wedge \neg Q) \vee P$
  - $P \wedge \neg Q \wedge R$
  - $\neg P \vee Q \vee R$
- non-exemples :
  - $P(\vee Q$
  - $P\neg Q$
  - $P \wedge \neg Q \vee R$

Syntaxe concrète

- parenthèses $\approx$ structure de l'arbre de syntaxe abstraite
- exemple : $((P \wedge \neg Q) \vee P)$
- exemples :
  - $(P \wedge \neg Q) \vee P$
  - $P \wedge \neg Q \wedge R$
  - $\neg P \vee Q \vee R$
- non-exemples :
  - $P(\vee Q$
  - $P\neg Q$
  - $P \wedge \neg Q \vee R$

Syntaxe concrète

- parenthèses $\approx$ structure de l'arbre de syntaxe abstraite
- exemple : $((P \wedge \neg Q) \vee P)$
- exemples :
  - $(P \wedge \neg Q) \vee P$
  - $P \wedge \neg Q \wedge R$
  - $\neg P \vee Q \vee R$
- non-exemples :
  - $P(\vee Q$
  - $P\neg Q$
  - $P \wedge \neg Q \vee R$

Syntaxe concrète

- parenthèses $\approx$ structure de l'arbre de syntaxe abstraite
- exemple : $((P \wedge \neg Q) \vee P)$
- exemples :
  - $(P \wedge \neg Q) \vee P$
  - $P \wedge \neg Q \wedge R$
  - $\neg P \vee Q \vee R$
- non-exemples :
  - $P(\vee Q$
  - $P\neg Q$
  - $P \wedge \neg Q \vee R$

Exercice

Implémenter la méthode Formule.toString() dans le fichier

`https://www.irif.fr/~schmitz/teach/2019_lo5/Formule.java`