Etude des tests conditionnels

Pourquoi la logique propositionnelle ?

Quiconque a écrit des programmes impératifs connaît les tests conditionnels.
Pourquoi les étudier à nouveau ?

- Toutes les logiques sont basées sur la logique propositionnelle dans une certaine mésure. On doit commencer par ici.
- Nous voulons gérer des tests conditionnels arbitrairement compliqués et étudier leur caractéristiques générales
- Nous ne voulons pas seulement évaluer les tests. Nous voulons aussi savoir quand deux tests signifient la même chose, quand un test implique un autre, si un test (ou un test de boucle) peut être jamais vrai ou faux.
- La logique propositionnelle réprésente une classe importante de problèmes computationnels. Comme ça, elle apparaît dans les algorithmes et dans la théorie de la complexité.

But : Spécifier quelles expressions sont considérées “bien formées”, ou “légales” indépendamment de leur signification et de leur vérité.

Ces expressions constitueront le langage formel de la logique propositionnelle.

1er Ingrédient

Les propositions atomiques

- Les symboles sont appelés *propositions atomiques*, ou *variables propositionnelles*.
- Ils ont la même fonction que des variables x, y, z en mathématiques.
- Mais comme ils sont propositionnels, nous utilisons d'habitude les lettres p, p', p_0, p_1, p_2, ..., ainsi que q, r, s, ...

2ème Ingrédient

Les connecteurs booléens

Nous sommes intéressés par les opérations booléennes suivantes, ou connecteurs :

- **et** : écrit comme $\wedge$ (ou parfois &, et dans les livres les plus vieux, '$\cdot$')
- **non** : écrit comme $\neg$ (ou parfois $\sim$)
- **ou** : écrit comme $\vee$ (dans les livres les plus vieux, $+$)
- **si alors**, ou 'implique'. Ceci est écrit comme $\rightarrow$ (ou parfois $\supset$), mais pas comme $\Rightarrow$, qui est utilisé différemment).
- **si et seulement si** : écrit comme $\leftrightarrow$ (mais pas $\Leftrightarrow$ ni $\equiv$, qui sont utilisés différemment)
- **vrai** et **faux** : écrit $\top$, $\bot$

3ème Ingrédient

La ponctuation

On a besoin des parenthèses pour éliminer l'ambiguïté de nos tests conditionnels.

C'est la même choses que les termes dans l'arithmétique : $1 - 2 + 3$ est ambigu, on peut le lire comme $(1 - 2) + 3$ ou $1 - (2 + 3)$, et la différence est importante.

Par exemple, $p \wedge q \vee r$ peut être lu comme

- $(p \wedge q) \vee r$
- $p \wedge (q \vee r)$

et la différence compte.

Donc, nous commençons par mettre tous les paranthèses dans les formulas, et après on supprime celles qui ne sont pas nécessaires.

Definition (Alphabet)

Un *alphabet* $\mathcal{A}$ d'un langage propositionnel $\mathcal{L}_P$ est l'union des ensembles suivants :

- Un ensemble dénombrable $P = \{p, q, r, ...\}$ de symboles dits *lettres (ou variables) propositionnelles*.
  Cet ensemble est aussi dit *signature* de $\mathcal{L}_P$.

- Un ensemble de symboles dits *connecteurs logiques*, contenant deux connecteurs à 0 arguments $\top$ et $\bot$, aussi dits *constantes propositionnelles*, un connecteur à un argument $\neg$ et quatre connecteurs binaires : $\wedge, \vee, \rightarrow, \leftrightarrow$.

- Les parenthèses : une ouvrante, (, et une fermante,).

Les Formules

Une formule propositionnelle est une séquence de symboles construite à partir des atomes propositionnels, en utilisant les connecteurs booléens ci-dessus, et les parenthèses, selon la façon appropriée.

Plus précisément :

Definition (Formules)

Les expressions bien formées ou *formules* d'un langage propositionnel $\mathcal{L}_P$ sont les mots sur l'alphabet de $\mathcal{L}_P$ qui sont éléments de l'ensemble défini par récurrence comme suit.

- Toute proposition atomique $p \in P$ est une formule.
  Aussi, les constantes propositionnelles $\top$ et $\bot$ sont des formules.
- si F est une formule, alors $(\neg F)$ est aussi une formule ;
- si F_1 et F_2 sont des formules, alors $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \rightarrow F_2)$, et $(F_1 \leftrightarrow F_2)$ sont aussi des formules.

Dit autrement :

$$F := p \mid \top \mid \bot \mid (\neg F) \mid (F * F)$$

où $p \in P$ et $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Exemples de Formules

Celles-ci sont des formules :

- p
- $(\neg p)$
- $((\neg p) \wedge \top)$
- $(\neg((\neg p) \wedge \top))$
- $((\neg p) \rightarrow (\neg((\neg p) \wedge \top)))$

On peut déjà voir qu'on a du mal avec les parenthèses.
Nous avons besoin de conventions pour nous débarrasser (de certaines).

Nous obtiendrons (à parler proprement) des *abréviations* de véritables formules.
Mais la plupart des gens pense à elles comme à des vraies formules.

On peut toujours omettre les parenthèses les plus extérieures :
$\neg p$ et $(\neg p) \rightarrow (\neg((\neg p) \wedge \top))$ ne sont pas ambigus.

Conventions pour simplifier l'écriture des formules

- Pas d'écriture des parenthèses les plus extérieures.
- Le connecteur $\neg$ s'applique à la plus petite formule le suivant immédiatement ; par ex. on pourra écrire $\neg p \wedge q$ à la place de $(\neg p) \wedge q$;
- $\wedge$ et $\vee$ sont prioritaires sur $\rightarrow$ et $\leftrightarrow$; par ex., on pourra écrire $p \wedge q \rightarrow r \vee p$ à la place de $(p \wedge q) \rightarrow (r \vee p)$ (analogie avec $\times$ par rapport à $+$ en arithmetique).
- si * est un connecteur binaire donné et $A_1, ..., A_n$, oú $n \geq 3$, sont des formules, on pourra écrire $A_1 * A_2 ... * A_n$ à la place de $(...(A_1 * A_2)...) * A_n$ (association à gauche).

Conventions pour simplifier l'écriture des formules

Pour eliminer plus de parenthèses, nous classons les connecteurs booléens selon leur force de liaison décroissante :

$$\text{(plus fort) } \neg, \wedge, \vee, \rightarrow, \leftrightarrow \text{ (plus faible)}$$

Il est comme dans l'arithmétique, où $\times$ est plus forte que $+$.
$2 + 3 \times 4$ est généralement lu comme $2 + (3 \times 4)$, pas comme $(2 + 3) \times 4$.
Donc :

- $p \vee q \wedge r$ est lu comme $p \vee (q \wedge r)$, pas comme $(p \vee q) \wedge r$
- $\neg p \wedge q$ est lu comme $(\neg p) \wedge q$, pas comme $\neg(p \wedge q)$
- $p \wedge \neg q \rightarrow r$ est lu comme $(p \wedge (\neg q)) \rightarrow r$, plutôt que $p \wedge (\neg(q \rightarrow r))$ ou $p \wedge ((\neg q) \rightarrow r)$

Arbre Syntaxique

Nous avons montré comment lire une formule sans ambiguïté, comment l'analyser. Cette information peut être représentée comme un arbre : l'arbre syntaxique de la formule.

Definition (Arbre Syntaxique)

L'*arbre syntaxique* d'une formule A, noté $Arbre_A$, est un arbre binaire $\langle G, R, D \rangle$ défini par récurrence sur la structure de A :

- Si A est atomique, son arbre syntaxique est $\langle \emptyset, A, \emptyset \rangle$.
- Si A est $(\neg A_1)$, alors son arbre syntaxique est $\langle Arbre_{A_1}, \neg, \emptyset \rangle$, où $Arbre_{A_1}$ est l'arbre syntaxique de A_1.
- Si A est $(A_1 \ * \ A_2)$ où $*$ est un connecteur binaire, alors son arbre syntaxique est $\langle Arbre_{A_1}, *, Arbre_{A_2} \rangle$, où $Arbre_{A_1}$ est l'arbre syntaxique de A_1 et $Arbre_{A_2}$ est l'arbre syntaxique de A_2.

Connecteur Principal

Definition (Connecteur Principal)

Si A contient au moins un connecteur, on appelle *connecteur principal* de A le connecteur qui se trouve à la racine de $Arbre_A$

Chaque formule non-atomique a un connecteur principal qui détermine sa *forme logique globale*. Vous devez apprendre à la reconnaître.

- $p \wedge q \rightarrow r$ a connecteur principal $\rightarrow$. Sa forme logique globale est $A \rightarrow B$.
- $\neg(p \rightarrow \neg q)$ a connecteur principal $\neg$. Sa forme logique globale est $\neg A$.
- $p \wedge q \wedge r$ a connecteur principal $\wedge$ (le 2ème !). Sa forme logique est $A \wedge B$.
- $p \vee q \wedge r$ a connecteur principal $\vee$. Sa forme logique est $A \vee B$.

Sous-formules

L'arbre syntaxique permet d'expliquer la notion de *sous-formule*.

Les sous-formules d'une formule A sont les formules construites dans les étapes de construction de A.

Elles correspondent à des noeuds, ou à des sous-arbres, de l'arbre syntaxique de A.

Definition (Sous-formules)

- Si A est une formule atomique, alors $sousf(A)$ est $\{A\}$;
- Si le connecteur principal de A est $\neg$, c'est-à-dire que A a la forme $\neg A_1$, alors $sousf(A)$ est l'union ensembliste de $\{\neg A_1\}$ et de $sousf(A_1)$
- Si le connecteur principal de A est un connecteur binaire *, c'est-à-dire que A a la forme $A_1 * A_2$, alors $sousf(A)$ est l'union ensembliste de $\{A_1 * A_2\}$, avec $sousf(A_1)$ et $sousf(A_2)$.

Termes techniques pour les formules logiques

Definition

- Une formule de la forme $\top$, $\bot$, ou p, pour un atome p, est appelée *atomique*.
- Une formule dont la forme logique est $\neg A$ est appelée une formule *niée*.
- Une formule de la forme $\neg p$, $\neg\top$, ou $\neg\bot$ est parfois appelé *niée-atomique*.
- Une formule de la forme $A \wedge B$ est appelée une *conjonction*, et A, B sont ses *conjoints*.
- Une formule de la forme $A \vee B$ est appelée une *disjonction*, et A, B sont ses *disjoints*.
- Une formule de la forme $A \to B$ s'appelle une *implication*. A est appelée l'*antécédente*, B est appelée *conséquente*.

Termes techniques pour les formules logiques (suite)

Definition

- Une formule qui est soit atomique soit niée-atomique est appelée *littéral* (en mathématiques on l'appelle parfois une formule de base).
- Une *clause* est une disjonction ($\vee$) d'un ou plusieurs littéraux.

Par exemple, le formules p, $\neg r$, $\neg\bot$, $\top$ sont tous des littéraux.

Quatre exemples de clauses

$$p$$
$$\neg p$$
$$p \vee \neg q \vee r$$
$$p \vee p \vee \neg p \vee \neg\bot \vee \top \vee \neg q$$

Certains considèrent la clause vide (la disjonction de zéro littéraux !), qui par convention est traitée de la même manière que $\bot$.

Conclusions

- Logique Classique Propositionnelle : syntaxe
- Formules : atomes, connecteurs et parenthèses
- Conventions d'Ecriture
- Arbre Syntaxique : connecteur principal, sous-formules