# LIFLC – Logique classique

## CM2 – Induction

Licence informatique UCBL – Automne 2018–2019

https://liris.cnrs.fr/ecoquery/dokuwiki/doku.php?id=enseignement:logique:start

Dpt Informatique

# Structures et règles de construction

De nombreuses structures en informatique se créent
avec des *règles de construction*

- listes, arbres
- nombres entiers
- ...

Les règles expliquent comment créer une structure plus grande à partir de structures plus petites

- Que définit-on ainsi ?
- Comment calculer sur ces structures ?
- Peut-on prouver des propriétés ?

# Ensembles inductifs

### Éléments de définition

- Des règles de construction
- L'ensemble est défini comme ***LE plus petit*** ensemble ***stable par les règles de constructions***

# Exemple : listes finies

Règles de construction :

- la *liste vide* [] est une liste finie
- si $l$ est une liste, $cons(h, l)$ est aussi une liste finie

Remarques :

- tout ensemble stable par ces règles contient la liste vide
- on a pas posé de contraintes sur $h$ ici :
  1 élément peut servir de base à de nombreux autres

# Exemple : entiers naturels

Règles de construction :

- 0 est un entier naturel
- si $n$ est un entier naturel alors $n+1$ est un entier naturel

Remarques :

- $\mathcal{N}$ est le plus petit ensemble stable par ces règles
- $\mathcal{Z}$ est également stable, mais est plus grand
  - certaines propriétés de $\mathcal{N}$ ne sont pas valables pour $\mathcal{Z}$
    ⚠ il est important de prendre le plus petit ensemble

# Exemple : arbres finis

Règles de construction :

- l'arbre vide $\square$ est un arbre
- si $a_1$ et $a_2$ sont des arbres, alors $node(h, a_1, a_2)$ est un arbre

Remarques :

- on est pas limité à l'utilisation d'un seul élément pour construire d'autres éléments

# Formules de logique propositionnelle

Une formule est constituée de *variables* ($p, p', q, \ldots$) et de *connecteurs* ($\neg, \vee, \wedge, \Rightarrow$)

Règles de construction :

- Les variables sont des formules
- Si $A$ est une formule, alors $(\neg A)$ est une formule
- Si $A$ et $B$ sont des formules alors $(A \vee B)$, $(A \wedge B)$ et $(A \Rightarrow B)$ sont des formules.

Remarque : on est pas limité à une seule règle de construction pour la fabrication d'un élement à partir d'éléments existants.

# Fonction récursive : longueur

Soit $l$ une liste

$$longueur(l) = \begin{cases} 0 & \text{si } l = [] \\ longueur(l') + 1 & \text{si } l = cons(h, l') \end{cases}$$

Remarque :

- la définition de *longueur* fait référence à *longueur*

# Fonctions récursives : principe

Définition divisée en cas :

- Des *cas de base* :
  - pas de référence à la fonction dans la définition
- Des *cas récursifs* :
  - La valeur de la fonction pour le paramètre courant est obtenue par un calcul sur la/les valeur(s) de la fonction pour un/d' autre(s) paramètres.

⚠ Il faut s'assurer de ne pas tourner en round ⚠

# Définition récursive mal formée

Soit la définition suivante :

$$pair(n) = \begin{cases} vrai & \text{si } pair(n+1) = faux \\ faux & \text{si } pair(n+1) = vrai \end{cases}$$

- La définition ne fixe pas la valeur que doit prendre la fonction.
- Intuitivement :
  - il manque un cas de base ;
  - une implémentation ferait appel à des entiers de plus en plus grands $\rightarrow$ pas d'arrêt.

# Fonctions récursives et ordres bien fondés

Comment s'assurer qu'une définition récursive $f$ est bien formée ?

Idée : Ordre bien fondé $\rightarrow$ pas de chaîne infinie strictement décroissante.

- on peut munir $dom(f)$ d'un ordre bien fondé
- tel qu'on assure que chaque référence à $f$ dans un cas récursif se fasse avec un paramètre strictement plus petit v-à-v de cet ordre

# Retour sur la fonction longueur

$$longueur(l) = \begin{cases} 0 & \text{si } l = [] \\ longueur(l') + 1 & \text{si } l = cons(h, l') \end{cases}$$

- Ordre $\leq$ sur les listes : fermeture réflexive et transitive de $\lhd$ où $l' \lhd l$ si et seulement si il existe $h$ tel que $l = cons(h, l')$.
- On peut remarquer que dans la définition, $l' < l$
- *longueur* est bien définie, par exemple :
  - $longueur(cons(a, cons(b, cons(c, []))))$ dépend de
  - $longueur(cons(b, cons(c, [])))$ qui dépend de
  - $longueur(cons(c, []))$ qui dépend de
  - $longueur([])$ qui vaut 0
- Les paramètres des appels récursifs “imbriqués” forment une séquence strictement décroissante.
  - Le dernier élément de la séquence correspond forcément à un cas de base.

# Règles de construction pour déconstruire

2 utilisations des règles de construction :

- Dans la définition d'ensemble inductif :
  les règles servent à construire une valeur
- Dans la définition d'une fonction récursive :
  les règles servent à *déconstruire* une valeur

La déconstruction sert à effectuer le calcul / à montrer la propriété désirée

# Exemple : profondeur d'un arbre

$$profondeur(a) = \begin{cases} 1 & \text{si} \quad a = \square \\ 1 + max(profondeur(a_1), profondeur(a_2)) & \\ & \text{si} \quad a = node(h, a_1, a_2) \end{cases}$$

Écrire $a = node(h, a_1, a_2)$ est une déconstruction de $a$
C'est aussi le cas pour $a = \square$

Ordre bien fondé sur les arbres similaire à celui sur les listes :

- $\leq$ est la fermeture réflexive transitive de $\lhd$ où
  si $a = node(h, a_1, a_2)$ alors $a_1 \lhd a$ et $a_2 \lhd a$

# Comment démontrer des propriétés sur des ensembles inductifs ?

Ensembles inductifs :

- Ensembles infinis
- Dont la description repose sur des règles de construction

Idée :

- Utiliser ces règles de construction pour structurer la preuve
- Déconstruire les valeurs comme pour les fonctions récursives

# Sur les entiers naturels : récurrence

- Objectif : démontrer une propriété
    - pour tous les entiers naturels

- Moyen :
    - Cas de base : prouver pour 0
    - Cas de récurrence : prouver que
        - si vrai pour $n$ (hypothèse de récurrence)
        - alors vrai pour $n+1$

Rmq : Parfois il faut faire la preuve pour quelques entiers fixés en plus de 0 et du passage de $n$ à $n+1$

# Sur les entiers naturels : récurrence

- Objectif : démontrer une propriété
  - pour tous les entiers naturels
- Moyen :
  - Cas de base : prouver pour 0 $\leftarrow$ attention à ne pas oublier
  - Cas de récurrence : prouver que
    - si vrai pour $n$ (hypothèse de récurrence)
    - alors vrai pour $n+1$

Rmq : Parfois il faut faire la preuve pour quelques entiers fixés en plus de 0 et du passage de $n$ à $n+1$

# Un exemple décortiqué

**Theorem**

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

- Cas de base : $n = 0$
  Il suffit de vérifier l'équation
- Cas de récurrence : On suppose que c'est vrai pour $n$ :

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

en ajoutant $n+1$ de chaque côté :

$$n+1+\sum_{i=0}^{n} i = n+1+\frac{n(n+1)}{2}$$

$$\sum_{i=0}^{n+1} i = \frac{2(n+1)+n(n+1)}{2}$$

$$\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$$

# Liens avec la récursivité

## Intuition

*Preuve par récurrence*

$\leftrightarrow$

*Procédure récursive de fabrication de preuve*

- Même fonctionnement :
  - condition d'arrêt $\leftrightarrow$ cas de base
  - appel récursif $\leftrightarrow$ utilisation de l'hypothèse de récurrence
- pour $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ :

  preuve pour $n = 2 \rightsquigarrow$ preuve pour $n = 1 \rightsquigarrow$ preuve pour $n = 0$

# Principe des démonstrations par inductions

Pour chaque règle de construction, montrer que :

- Si la propriété est vraie pour chaque élément utilisé par la règle
  - **hypothèse d'induction**
- alors elle est vraie pour ce qui est fabriqué par la règle

# Exemple abstrait sur les listes

On peut montrer qu'une propriété $P$ est vraie sur les listes.

- On montre que $P$ est vraie pour la liste vide []
  - Il n'y a pas d'élément utilisé par la règle de la liste vide
    $\rightarrow$ il faut montrer $P$ directement pour ce cas
- On montre que $P$ est vraie pour une liste $l = cons(h, l')$ :
  - si $P$ est vraie pour $l'$ (hypothèse d'induction)
  - alors $P$ est vraie pour $l$

# Fonctionnement

Ce type de démonstration est-il correct ?

Ce qu'on montre exactement : l'ensemble des éléments vérifiant $P$ est stable par les règles de construction.

Comme l'ensemble inductif défini par ces règles est LE plus petit de tous ces ensembles, il est inclus dans l'ensemble des éléments vérifiant $P$, donc tous ses éléments vérifient $P$.

# Fonctionnement - 2

Preuve par induction

$\leftrightarrow$

Procédure récursive de fabrication de preuve

- À partir des preuves pour les éléments utilisés par la règle de construction,
- on explique comment faire la preuve pour l'élément produit par la règle

# Exemple concret sur les arbres

**Theorem**

*Un arbre (binaire) $A$ à $n$ feuilles (ici des arbres vides $\square$) comporte $n-1$ nœuds internes*

- Vrai si l'arbre est réduit à une feuille (*i.e.* $A = \square$)
- Sinon sa racine est un nœud interne et ils a deux fils $A_1$, $A_2$ (*i.e.* $A = node(h, A_1, A_2)$)

$A_1$ a $n_1$ feuilles et $A_2$ a $n_2$ feuilles

Par hypothèse d'induction (car $A_1$ et $A_2$ sont des composants de $A$) :

$A_1$ a $n_1 - 1$ nœuds internes
et $A_2$ en a $n_2 - 1$.

Nombre de noeuds internes de $A$ :

$$\underbrace{n_1 + n_2}_{\#\text{ feuilles } A} - 1 - 1 + 1$$