Errata à l'édition 2004-2005 du syllabus d'Algorithmique Générale II

3 août 2005

1 Partie théorique et énoncés

Étant donné que le cours se donne maintenant en BA2, son titre et sa mnémonique ont changé. Il faut donc remplacer, dans tout le syllabus, « Algorithmique Générale II » par « Algorithmique II » et « INFO091 » par « INFO-F-203 ».

Dans la suite de ce document, les changements par rapport à la version précédente du syllabus ont été encadrés.

Page	Correction
i	Le premier paragraphe de l'avertissement devient : Ce document représente la seconde édition du syllabus d'exercices pour le cours d'Algorithmique II (anciennement Algorithmique Générale II). Cette édition se base sur un premier syllabus réalisé en 2002–2003 et 2003–2004 par Gilles Geeraerts et Steve Kremer, et distribué aux étudiants *via* le Web (avec, séance par séance, les corrigés des exercices).
5	Le premier paragraphe après l'encadré devient : Cette façon de procéder est tout à fait systématique, car on aura décrit le comportement de chaque opération quelque soit l'instance légale de la structure de données qui est passée à l'opération. En effet, chaque instance peut, répétons-le, être décrite en terme de constructeurs minimaux, et le comportement de chaque opération est décrit sur les constructeurs minimaux.
6	Dans la Fig. 2.2, le dernier axiome devient : $\boxed{\text{pop}}(\underline{\text{push}}(p, e)) = p$
8	Dans l'exercice **A.5**, le second paragraphe devient : Démontrez ensuite (à l'aide de vos axiomes) que l'$\boxed{\text{union}}$ d'un ensemble...

Page	Correction
11	– La première ligne devient : Une *fonction* f de l'ensemble A vers l'ensemble B est une relation, qui, à chaque élément de A... – La deuxième paragraphe de **Définition(s) d'un graphe** devient : Dans le cas d'un graphe *non-dirigé* : nous pouvons simplement faire en sorte que, chaque fois qu'un arc (s, d) appartient à E,... – Le troisième paragraphe de la même section devient : Pour obtenir un graphe *pondéré*, nous pouvons ajouter au graphe une fonction de poids $W : E \mapsto D$, où D est le domaine dans lequel on choisira les pondérations (il s'agit typiquement de $\mathbb{R}$ ou $\mathbb{N}$). L'expression $W(i, j)$ nous fournit le poids de l'arc (i, j)...

2 Types de données abstraits – corrigés

Slide	Correction
3	Remplacer $\underline{\text{FileVide}} :\longrightarrow \mathsf{File}$ par $\underline{\text{FileVide}} :\emptyset \rightarrow \mathsf{File}$
11	Les axiomes ont été numérotés. La démonstration du théorème a été ajoutée. Le nouveau slide 11 ainsi que les slides donnant la démonstration sont fournis en annexe.
13	Même type de modifications. Les nouveaux slides sont fournis en annexe (slides 16 et suivants)
16	La correction du `CHeap` a été ajoutée et est fournie en annexe (slides 21 et suivants)

3 Graphes – corrigés

Slide	Correction
11	Dans la fonction `Clean`, la ligne $a \leftarrow a.\text{arc\_svt}$; doit se trouver en-dehors du **else** mais doit rester à l'intérieur du **while**.
25	La première ligne du **if** $\ell < best$ **then** doit être $best \leftarrow \ell$;.
32	L'algorithme de Ford & Fulkerson a été ajouté et est fourni en annexe (slides 32 à 34).

4 Dérécursification – corrigés

Slide	Correction
13–14	Un slide a été ajouté entre ces deux slides pour compléter la dérécursification de la fonction factorielle. Il est fourni en annexe (slide 14).
27	La condition du **while** englobant est $\neg S.\textbf{empty}() \vee \mathit{appel}$.

Exercice A.5 – L'Ensemble

Sorte

Ensemble

Utilise

Booléen, Élément, Entier

Opérations

- $\underline{\text{Vide}}$: $\rightarrow$ Ensemble
- $\underline{\text{Ajouter}}$: Élément $\times$ Ensemble $\rightarrow$ Ensemble
- Retirer: Élément $\times$ Ensemble $\rightarrow$ Ensemble
- $\cup$: Ensemble $\times$ Ensemble $\rightarrow$ Ensemble
- $\cap$: Ensemble $\times$ Ensemble $\rightarrow$ Ensemble
- $\in$: Élément $\times$ Ensemble $\rightarrow$ Booléen
- $\#$: Ensemble $\rightarrow$ Entier

Préconditions

aucune

Axiomes

1. $\in(x,\underline{\text{Vide}}) = \text{faux}$
2. **si** $x = y$ **alors** $\in(x,\text{Ajouter}(y,e)) = \text{vrai}$
3. **si** $x \neq y$ **alors** $\in(x,\text{Ajouter}(y,e)) = \in(x,e)$
4. $\#(\underline{\text{Vide}}) = 0$
5. $\#(\underline{\text{Ajouter}}(x,e)) = \#(e)$ **ssi** $\in(x,e)$
6. $\#(\underline{\text{Ajouter}}(x,e)) = \#(e) + 1$ **ssi** $\neg \in(x,e)$

7. $\text{Retirer}(x,\underline{\text{Vide}}) = \underline{\text{Vide}}$
8. $\text{Retirer}(x,\underline{\text{Ajouter}}(y,e)) =$
   $\underline{\text{Ajouter}}(\text{Retirer}(x,e),y)$ **ssi** $x \neq y$
9. $\text{Retirer}(x,\underline{\text{Ajouter}}(y,e)) = e$ **ssi** $\neg\in(x,e) \wedge x = y$
10. $\text{Retirer}(x,\underline{\text{Ajouter}}(y,e)) = \text{Retirer}(x,e)$ **ssi** $\in(x,e)$
    $\wedge x = y$
11. $\cup(e_1, \underline{\text{Vide}}) = e_1$
12. $\cup(\underline{\text{Vide}}, e_2) = e_2$
13. $\cup(\underline{\text{Ajouter}}(x,e_1),e_2) = \underline{\text{Ajouter}}(x,\cup(e_1,e_2))$
14. $\cup(e_1, \underline{\text{Ajouter}}(x,e_2)) = \underline{\text{Ajouter}}(x,\cup(e_1,e_2))$
15. $\cap(e_1, \underline{\text{Vide}}) = \underline{\text{Vide}}$
16. $\cap(\underline{\text{Vide}}, e_2) = \underline{\text{Vide}}$
17. $\cap(e_1, \underline{\text{Ajouter}}(x,e_2)) = \underline{\text{Ajouter}}(x,\cap(e_1,e_2))$
    ssi $\in(x,e_1)$
18. $\cap(e_1, \underline{\text{Ajouter}}(x,e_2)) = \cap(e_1,e_2)$ **ssi** $\neg\in(x,e_1)$
19. $\cap(\underline{\text{Ajouter}}(x,e_1),e_2) = \underline{\text{Ajouter}}(x,\cap(e_1,e_2))$
    ssi $\in(x,e_2)$
20. $\cap(\underline{\text{Ajouter}}(x,e_1),e_2) = \cap(e_1,e_2)$ **ssi** $\neg\in(x,e_2)$

Théorème 1 *Si e_1 et e_2 sont deux ensembles tels que : $\#(e_1) = n$ et $\#(e_2) = n$, alors : $\#\big(\cup(e_1, e_2)\big) \leq m + n$.*

Preuve. Par induction sur la taille des ensembles

- Cas de Base. e_1 =Vide et e_2 =Vide. Par l'axiome 11 : $\cup(e_1, e_2)$=Vide. Par l'axiome 4, $\#(\text{Vide}) = 0 \leq 0 + 0$.
- Cas inductif. Hypothèse d'induction : Le théorème est vrai pour toute paire d'ensembles e_1, e_2 tels que : $\#(e_1) \leq k$ et $\#(e_2) \leq \ell$. Montrons qu'il est vrai si $\#(e_1) \leq k + 1$ et $\#(e_2) \leq \ell + 1$.

Comme l'union est commutative (*cfr.* axiomes 11–14), il suffit donc de prouver que le théorème est vrai si on augmente la taille d'un seul des deux ensembles : $\#(e_1) \leq k+1$ et $\#(e_2) \leq \ell$.
Supposons que $e_1 = \underline{\text{Ajouter}}(x, e)$. Et calculons le cardinal de $\cup(e, e_2)$. Il y a deux possibilités :

1. $\neg\in(x, e)$. Dans ce cas, on passe à la suite.
2. $\in(x, e)$. Dans ce cas :

$$\begin{aligned} & \#\Big(\cup\big(\underline{\text{Ajouter}}(x, e), e_2\big)\Big) \\ = \quad & \#\big(\cup(e, e_2)\big) \end{aligned}$$

On peut donc reprendre le raisonnement en remplaçant e_1 par e et recommencer jusqu'à se retrouver dans les conditions du point 1.

Maintenant on sait que $e_1 = \underline{\text{Ajouter}}(x, e)$, avec $\#(e) \leq k$. On a donc :

$$
\begin{array}{lll}
 & \#\big(\cup(e_1, e_2)\big) & \\
= & \#\Big(\cup\big(\underline{\text{Ajouter}}(x, e), e_2\big)\Big) & \\
= & \#\Big(\underline{\text{Ajouter}}\big(x, \cup(e, e_2)\big)\Big) & \text{axiome 13} \\
= & \#\big(\cup(e, e_2)\big) + 1 & \text{axiome 5} \\
\leq & k + \ell + 1 & \text{Par H.I.}
\end{array}
$$

□

Exercice A.6 – La Chaîne (de caractères)

Sorte

Chaîne

Utilise

Caractère, Booléen, Entier

Opérations

- Vide: $\emptyset \rightarrow$ Chaîne
- Ajoute: Chaîne $\times$ Caractère $\rightarrow$ Chaîne
- EPC: Chaîne $\rightarrow$ Chaîne
- Concat: Chaîne $\times$ Chaîne $\rightarrow$ Chaîne
- $EstVide$: Chaîne $\rightarrow$ Booléen
- $Longueur$: Chaîne $\rightarrow$ Entier
- $Car@$: Chaîne $\times$ Entier $\rightarrow$ Caractère
- $Egal$: Chaîne $\times$ Chaîne $\rightarrow$ Booléen

Préconditions

- $\text{EPC}(S)$ **est défini ssi** $\neg EstVide(S)$
- $Car@(S, i)$ **est défini ssi** $0 < i \leq Longueur(S)$

Axiomes

1. $\text{EPC}(\underline{\text{Ajoute}}(S,c)) = \underline{\text{Vide}}$ **ssi** $\mathit{EstVide}(S)$
2. $\text{EPC}(\underline{\text{Ajoute}}(S,c)) = \underline{\text{Ajoute}}(\text{EPC}(S),c)$ **ssi** $\neg \mathit{EstVide}(S)$
3. $\text{Concat}(\underline{\text{Vide}},\underline{\text{Vide}}) = \underline{\text{Vide}}$
4. $\text{Concat}(\underline{\text{Ajoute}}(S,c),\underline{\text{Vide}}) = \underline{\text{Ajoute}}(S,c)$
5. $\text{Concat}(\underline{\text{Vide}},\underline{\text{Ajoute}}(S,c)) = \underline{\text{Ajoute}}(S,c)$
6. $\text{Concat}(\underline{\text{Ajoute}}(S_1,c_1),\underline{\text{Ajoute}}(S_2,\ c_2)) =$
   $\underline{\text{Ajoute}}(\text{Concat}(\underline{\text{Ajoute}}(S_1,c_1),S_2),c_2)$
7. $\mathit{EstVide}(\underline{\text{Vide}}) = \text{vrai}$
8. $\mathit{EstVide}(\underline{\text{Ajoute}}(S,c)) = \text{faux}$
9. $\mathit{Longueur}(\underline{\text{Vide}}) = 0$
10. $\mathit{Longueur}(\underline{\text{Ajoute}}(S,c)) = \mathit{Longueur}(S)+1$
11. **si** $\mathit{Car@}(\underline{\text{Ajoute}}(S,c),\ \text{i}) = c$ **alors** $i = \mathit{Longueur}(S)+1$[a]

[a] ou bien : $i = \mathit{Longueur}(\underline{\text{Ajoute}}(S,c))$

12. **si** $\mathit{Car@}(\underline{\text{Ajoute}}(S, c), \text{i}) = \mathit{Car@}(S, i)$ **alors** $i < \mathit{Longueur}(S) + 1$
13. $\mathit{Egal}(\underline{\text{Vide}}, \underline{\text{Vide}}) = \text{vrai}$
14. $\mathit{Egal}(\underline{\text{Vide}}, \underline{\text{Ajoute}}(S, c)) = \text{faux}$
15. $\mathit{Egal}(\underline{\text{Ajoute}}(S, c), \underline{\text{Vide}}) = \text{faux}$
16. $\mathit{Egal}(\underline{\text{Ajoute}}(S_1, c_1), \underline{\text{Ajoute}}(S_2, c_2)) = \mathit{Egal}(S_1, S_2)$ **ssi** $c_1 = c_2$
17. $\mathit{Egal}(\underline{\text{Ajoute}}(S_1, c_1), \underline{\text{Ajoute}}(S_2, c_2)) = \text{faux}$ **si** $c_1 \neq c_2$

Théorème 2 *Pour toute chaîne de caractères s :* $\mathit{Egal}(s, s)$=*vrai.*

Preuve. Par induction sur la longueur de la chaîne.

Cas de base : $\mathit{Longueur}(s) = 0$

- Par l'axiome 9, on a : $s =$Vide.
- Par l'axiome 13, on a bien que Egal(Vide,Vide)=vrai.

Cas inductif : $\mathit{Longueur}(s) = \ell \geq 1$

- Hypothèse d'induction : le théorème est vrai pour toute chaîne de longueur $\leq \ell - 1$.
- Supposons que $s =$Ajoute(s', c). Par l'axiome 12, on sait que $\mathit{Longueur}(s') = \ell - 1$.
- Donc, par hypothèse d'induction : $\mathit{Egal}(s', s') =$ vrai.
- On veut montrer que :

$$\mathit{Egal}\big(\text{Ajoute}(s', c), \text{Ajoute}(s', c)\big) = \text{vrai}$$

- Par l'axiome 16, c'est équivalent à :

$$\mathit{Egal}\big(s', s'\big) = \text{vrai}$$

  Ce qui est vrai.

□

Théorème 3 *Si s_1 et s_2 sont deux chaînes de caractères telles que $Longueur(s_1) \neq Longueur(s_2)$, alors $Egal(s_1, s_2)$= faux.*

Preuve.

- Comme *Egal* est commutatif (preuve : exercice), on peut suppose que $Longueur(s_1) < Longueur(s_2)$.
- On fait la preuve par induction sur la longueur de s_1
  1. **Cas de Base** : $Longueur(s_1) = 0$.
     - Donc s_1=Vide (axiome 9).
     - Comme $Longueur(s_1) < Longueur(s_2)$, on peut exprimer s_2 sous la forme $Ajoute(s_3, c)$.
     - Axiome 14 : $Egal(Ajoute(s_3, c)), Vide)$=faux.
  2. **Cas inductif** : $Longueur(s_1) = \ell \geq 1$.
     - H.I. : le théorème est vrai quand $Longueur(s_1) < \ell$.
     - Comme s_1 et s_2 sont de longueur ≥ 1, on peut les exprimer sous la forme : s_1 =$Ajoute(s_3, c)$ et s_2 =$Ajoute(s_4, c')$.
     - On considère deux cas :
       - Soit $c \neq c'$. Dans ce cas : $Egal(Ajoute(s_3, c), Ajoute(s_4, c')) =$ faux, par l'axiome 17.
       - Soit $c = c'$. Dans ce cas : $Egal(Ajoute(s_3, c), Ajoute(s_4, c')) = Egal(s_3, s_4)$, par l'axiome 16. Or $Longueur(s_3) = \ell - 1$, par l'axiome 10. Donc $Egal(s_3, s_4) =$ faux, par H.I.

□

Exercice A.7 – Le CHeap

Type

CHeap

Utilise

Booléen, Élément, Couleur

Opérations

- Vide: $\emptyset \mapsto$ CHeap
- Insertion: CHeap $\times$ Iment $\times$ Couleur $\mapsto$ CHeap
- *Sommet*: CHeap $\times$ Couleur $\mapsto$ Iment
- *EstVide*: CHeap $\mapsto$ Boolen
- *EstCouleur*: CHeap $\times$ Couleur $\mapsto$ Boolen
- Supprime: CHeap $\times$ Couleur $\mapsto$ CHeap

Préconditions

- *Sommet* $(\mathcal{C}, c)$ est défini ssi *EstCouleur* $(\mathcal{C}, c)$
- Supprime $(\mathcal{C}, c)$ est défini ssi *EstCouleur* $(\mathcal{C}, c)$

Axiomes

1. $\mathit{Sommet}\left(\underline{\text{Insertion}}\left(\mathcal{C}, i, c\right), k\right) =$
$$\begin{cases} i & \text{si } c = k \text{ et } \neg \mathit{EstCouleur}\left(\mathcal{C}, c\right) \\ i & \text{si } c = k \text{ et } \mathit{EstCouleur}\left(\mathcal{C}, c\right) \text{ et } i < \mathit{Sommet}\left(\mathcal{C}, c\right) \\ \mathit{Sommet}\left(\mathcal{C}, k\right) & \text{sinon} \end{cases}$$
2. $\text{Supprime}\left(\underline{\text{Insertion}}\left(\mathcal{C}, i, c\right), k\right) =$
$$\begin{cases} \mathcal{C} & \text{si } c = k \text{ et } \neg \mathit{EstCouleur}\left(\mathcal{C}, c\right) \\ \mathcal{C} & \text{si } c = k \text{ et } \mathit{EstCouleur}\left(\mathcal{C}, c\right) \\ & \quad \text{et } i < \mathit{Sommet}\left(\mathcal{C}, c\right) \\ \underline{\text{Insertion}}\left(\text{Supprime}\left(\mathcal{C}, k\right), i, c\right) & \text{sinon} \end{cases}$$
3. $\mathit{EstVide}\left(\underline{\text{Vide}}\right) = \mathit{vrai}$
4. $\mathit{EstVide}\left(\underline{\text{Insertion}}\left(\mathcal{C}, i, c\right)\right) = \mathit{faux}$
5. $\mathit{EstCouleur}\left(\underline{\text{Vide}}, c\right) = \mathit{faux}$
6. $\mathit{EstCouleur}\left(\underline{\text{Insertion}}\left(\mathcal{C}, i, c\right), k\right) = \begin{cases} \mathit{vrai} & \text{si } c = k \\ \mathit{EstCouleur}\left(\mathcal{C}, k\right) & \text{sinon} \end{cases}$

– Le CHeap demandé est décrit comme suit :

$$\mathcal{C} = \underline{\text{Insertion}}\left(\underbrace{\underline{\text{Insertion}}\left(\overbrace{\underline{\text{Insertion}}\left(\underbrace{\underline{\text{Insertion}}\left(\underline{\text{Insertion}}\left(\underline{\text{Vide}}, 3, \circ\right), 2, \circ\right)}_{\mathcal{C}_1}, 1, \bullet\right)}^{\mathcal{C}_2}, 5, \bullet\right)}_{\mathcal{C}_3}, 2, \bullet\right)$$

	EstCouleur$(., \bullet)$	*EstCouleur*$(., \circ)$	*Sommet*$(., \bullet)$	*Sommet*$(., \circ)$
$\mathcal{C}_1$	faux	vrai	n.d.	2
$\mathcal{C}_2$	vrai	vrai	1	2
$\mathcal{C}_3$	vrai	vrai	1	2

– La première suppression :

$$
\begin{aligned}
& \mathrm{Supprime}\,(\underline{\mathrm{Insertion}}\,(\mathcal{C}_3, 2, \bullet)\,, \bullet) \\
= \;& \underline{\mathrm{Insertion}}\,(\mathrm{Supprime}\,(\underline{\mathrm{Insertion}}\,(\mathcal{C}_2, 5, \bullet)\,, \bullet)\,, 2, \bullet) && \text{axiome } 2-2 \\
= \;& \underline{\mathrm{Insertion}}\,(\underline{\mathrm{Insertion}}\,(\mathrm{Supprime}\,(\underline{\mathrm{Insertion}}\,(\mathcal{C}_1, 1, \bullet)\,, \bullet)\,, 5, \bullet)\,, 2, \bullet) && \text{axiome } 2-2 \\
= \;& \underbrace{\underline{\mathrm{Insertion}}\,(\underline{\mathrm{Insertion}}\,(\mathcal{C}_1, 5, \bullet)\,, 2, \bullet)}_{\mathcal{C}_4} && \text{axiome } 2-3
\end{aligned}
$$

– Par un raisonnement similaire...

$$
\mathrm{Supprime}\,(\mathcal{C}_4, \circ) = \underbrace{\underline{\mathrm{Insertion}}\,(\underline{\mathrm{Insertion}}\,(\underline{\mathrm{Insertion}}\,(\underline{\mathrm{Vide}}, 3, \circ)\,, 5, \bullet)\,, 2, \bullet)}_{\mathcal{C}_5}
$$

– Le sommet $\circ$:

$$
\begin{aligned}
& \mathit{Sommet}\,(\mathcal{C}_5, \circ) \\
= \;& \mathit{Sommet}\,(\underline{\mathrm{Insertion}}\,(\underline{\mathrm{Insertion}}\,(\underline{\mathrm{Vide}}, 3, \circ)\,, 5, \bullet)\,, \circ) && \text{axiome } 1-3 : \bullet \neq \circ \\
= \;& \mathit{Sommet}\,(\underline{\mathrm{Insertion}}\,(\underline{\mathrm{Vide}}, 3, \circ)\,, \circ) && \text{axiome } 1-3 : \bullet \neq \circ \\
= \;& 3 && \text{axiome } 1-1
\end{aligned}
$$

– $\mathcal{C}_5$ est-il vide ? Non par l'axiome 4

Exercice G.25

Ford-Fulkerson

void `MF(`***Graphe*** $G = \langle V, E \rangle$, ***Noeud*** s, ***Noeud*** t**)**
begin
    Ensemble $\mathcal{E}$;
    Noeud x ;
    foreach *arc* $a \in E$ **do** $a.\text{flow} \leftarrow 0$;
    while *true* **do**
        foreach *nœud* $v \in V$ **do** $v.\text{label} \leftarrow \perp$;
        $s.\text{label} \leftarrow (\perp, +\infty)$;
        $\mathcal{E} \leftarrow \{s\}$;
        Étape 1
        if $t.\text{label} = \perp$ **then break** ;
        else
            Étape 2
end

Étape 1

Cette étape permet de trouver un flux augmentant. Si, à la fin de l'étape, $t.\text{label} = \bot$, aucun flux augmentant n'a été trouvé.

begin
  while $\mathcal{E} \neq \emptyset \wedge t.\text{label} = \bot$ **do**
    Prendre x dans $\mathcal{E}$;
    $(z^{\pm}, \varepsilon(x)) \leftarrow x.\text{label}$;
    foreach *successeur* y *de* x **do**
      if $y.\text{label} = \bot \wedge (x,y).\text{flow} < (x,y).\text{capa}$ **then**
        $\varepsilon(y) \leftarrow \min\{\varepsilon(x), (x,y).\text{capa} - (x,y).\text{flow}\}$;
        $y.\text{label} \leftarrow (x^{+}, \varepsilon(y))$;
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{y\}$;
    foreach *prédécesseur* y *de* x **do**
      if $y.\text{label} = \bot \wedge (y,x).\text{flow} > 0$ **then**
        $\varepsilon(y) \leftarrow \min\{\varepsilon(x), (y,x).\text{flow}\}$;
        $y.\text{label} \leftarrow (x^{-}, \varepsilon(y))$;
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{y\}$;
end

Étape 2

Cette étape établit le flux qui a été trouvé à l'étape précédente.

begin
  $x \leftarrow t$;
  while $x \neq s$ **do**
    $(q, \varepsilon(x)) \leftarrow x.\text{label}$;
    if *q est de la forme y^+* **then**
      $(y, x).\text{flow} \leftarrow (y, x).\text{flow} + \varepsilon(t)$;
    else if *q est de la forme y^-* **then**
      $(x, y).\text{flow} \leftarrow (x, y).\text{flow} - \varepsilon(t)$;
    $x \leftarrow y$;
end

Version dérécursifiée ultime...

Si n_0 est la valeur de n avant la première boucle, son invariant est :

$$k + n = n_0$$

La post-condition est donc :

$$k = n_0 - 1 \wedge n = 1$$

Pour la seconde boucle :

$$k + n = n_0 \wedge \forall 0 \leq i \leq n : \texttt{fact}[i] = i!$$

La post-condition est :

$$n = n_0 \wedge \forall 0 \leq i \leq n : \texttt{fact}[i] = i!$$

Cela nous permet de simplifier (encore) la fonction...

void `factorielle(`***Integer*** n`)` **begin**
  Integer k ;
  $k \leftarrow n - 1$;
  $n \leftarrow 1$;
  $\texttt{fact}[n] \leftarrow 1$;
  while $n \leq k$ **do**
    $n \leftarrow n + 1$;
    $\texttt{fact}[n] \leftarrow n * \texttt{fact}[n - 1]$;
end