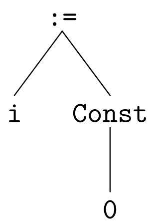
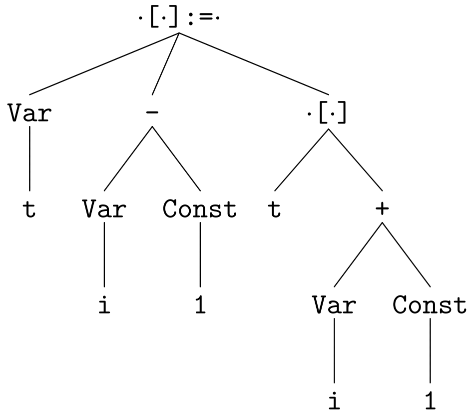
Corrigé de l'examen final de compilation

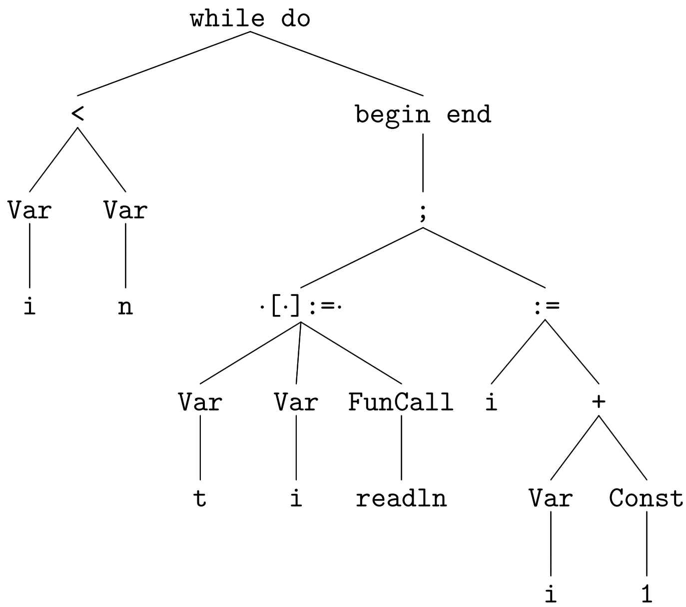
ÉNSIIE, semestre 3

mercredi 8 janvier 2014

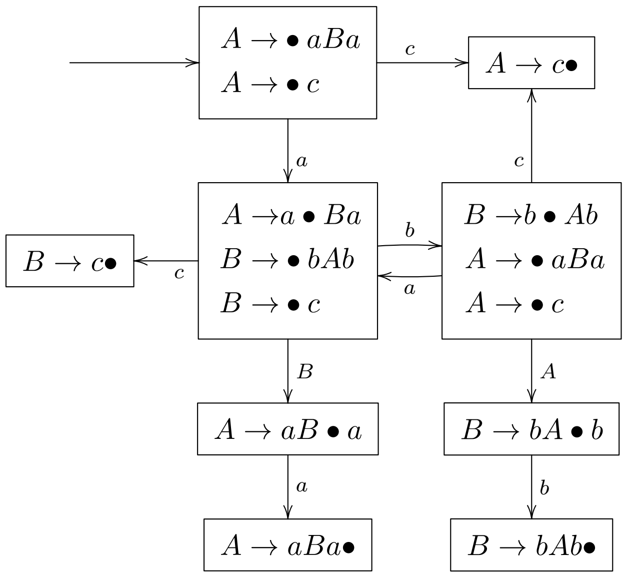
Exercice 1 : Syntaxe (2 points)

1. 

2. 

3. 

Exercice 2 : Analyse syntaxique (3 points)

1. 

2. Il n'y a ni conflit réduire/réduire, ni conflit décaler/réduire, la grammaire est LR(0).

Exercice 3 : Réécriture (5 points)

1. x
2. Chaque règle fait décroître le nombre de symboles, et il n'y a pas de variables dupliquées à droite, donc chaque pas de réécriture diminue strictement le nombre de symbole du terme. Le système est donc fortement normalisant.
3. Les paires critiques sont les suivantes :
   - $\texttt{undo}(x)$ et $\texttt{undo}(x)$ en partant de $\texttt{undo}(\texttt{undo}(\texttt{undo}(x)))$ et en appliquant la première règle à deux positions différentes.
   - $\texttt{compose}(x, \texttt{undo}(x))$ et $\texttt{noop}$ en partant de $\texttt{compose}(\texttt{undo}(\texttt{undo}(x)), \texttt{undo}(x))$ en appliquant la première et la deuxième.

   Il n'y a pas d'autre superposition possibles.

   La première paire critique est joignable, pas la seconde.
4. On oriente en $\texttt{compose}(x, \texttt{undo}(x)) \to \texttt{noop}$. On a les nouvelles paires critiques suivantes :
   - $\texttt{compose}(\texttt{undo}(x), x)$ et $\texttt{noop}$ avec la première en partant de $\texttt{compose}(\texttt{undo}(x), \texttt{undo}(\texttt{undo}(x)))$
   - $\texttt{undo}(\texttt{noop})$ et $\texttt{noop}$ avec la troisième en partant de $\texttt{compose}(\texttt{noop}, \texttt{undo}(e))$.

   La première paire critique est joignable (grâce à la seconde règle de réécriture).

   La deuxième paire critique n'est pas joignable.
5. On ajoute $\texttt{undo}(\texttt{noop}) \to \texttt{noop}$. On a les nouvelles paires critiques suivantes :
   - $\texttt{noop}$ et $\texttt{undo}(\texttt{noop})$ avec la première en partant de $\texttt{undo}(\texttt{undo}(e))$
   - $\texttt{noop}$ et $\texttt{compose}(\texttt{noop}, \texttt{noop})$ avec la seconde en partant de $\texttt{compose}(\texttt{undo}(\texttt{noop}), \texttt{noop})$
   - $\texttt{undo}(\texttt{noop})$ et $\texttt{compose}(\texttt{noop}, \texttt{noop})$ avec la troisième en partant de $\texttt{compose}(\texttt{noop}, \texttt{undo}(\texttt{noop}))$
   - $\texttt{noop}$ et $\texttt{compose}(\texttt{noop}, \texttt{noop})$ avec la quatrième en partant de $\texttt{compose}(\texttt{noop}, \texttt{undo}(\texttt{noop}))$.

   Toutes ces paires critiques sont joignables. De plus, le système formé des cinq règles est toujours fortement normalisant. Par conséquent il est confluent

Exercice 4 : Convention d'appel (5 points)

1. Registres callee-saved (`$ra $s0 ... $s7`) : `hanoi` fait des appels de fonction. Par conséquent elle modifie `$ra`. Elle modifie aussi `$s0` (variable locale k), mais pas les autres `$si`. Pour préserver les registres `$ra $s0` lors de l'appel, il faut les sauvegarder.

   Registres caller-saved (`$a0 ... $a3 $t0 ... $t9`) `hanoi` ne modifie pas `$a3 $t0 ... $t9`. Elle modifie `$a0 $a1 $a2` lors des appels de fonction. `$a0 $a1 $a2`, qui contiennent `n i j`, doivent être sauvegardés avant l'appel à `other(i,j)` car on a besoin de leur valeur après (même si la valeur de `$a2` n'est pas modifiée par le passage des argument, elle pourrait être modifiée par `other` au cours de l'appel).

   La trame de `hanoi` est donc :

sauvegarde de `$ra`
sauvegarde de `$s0`
sauvegarde de `$a0`
sauvegarde de `$a1`
sauvegarde de `$a2`

2. `other` ne fait pas d'appel et ne modifie donc pas `$ra`. Il n'a pas de variable locale, donc `other` ne modifie pas `$s0 ... $s7 $t0 ... $t9`. Il n'y a pas d'appel, donc pas besoin de sauvegarder les registres caller-saved. Au final, `other` n'a besoin de sauvegarder aucun registre. Sa trame est donc vide.

3.

	←`$sp`

`$a0`←?
`$a1`←?
`$a3`←?
`$v0`←?
`$ra`←?
`$s0`←?

	←`$sp`

`$a0`←2
`$a1`←1
`$a3`←3
`$v0`←?
`$ra`←23
`$s0`←?

	←`$sp`

`$a0`←2
`$a1`←1
`$a3`←3
`$v0`←?
`$ra`←23
`$s0`←?

23	
?	
	←`$sp`

`$a0`←2
`$a1`←1
`$a3`←3
`$v0`←?
`$ra`←23
`$s0`←?

23	
?	
2	
1	
3	←`$sp`

`$a0`←2
`$a1`←1
`$a3`←3
`$v0`←?
`$ra`←23
`$s0`←?

23	
?	
2	
1	
3	←`$sp`

`$a0`←1
`$a1`←3
`$a3`←3
`$v0`←?
`$ra`←16
`$s0`←?

23	
?	
2	
1	
3	←`$sp`

`$a0`←1
`$a1`←3
`$a3`←3
`$v0`←2
`$ra`←16
`$s0`←?

23	
?	
2	
1	
3	←`$sp`

`$a0`←2
`$a1`←1
`$a3`←3
`$v0`←2
`$ra`←16
`$s0`←?

23	
?	
2	
1	
3	←`$sp`

`$a0`←2
`$a1`←1
`$a3`←3
`$v0`←2
`$ra`←16
`$s0`←2

23	
?	
2	
1	
3	←`$sp`

`$a0`←1
`$a1`←2
`$a3`←3
`$v0`←2
`$ra`←17
`$s0`←2

23	
?	
2	
1	
3	
	←`$sp`

`$a0`←1
`$a1`←2
`$a3`←3
`$v0`←2
`$ra`←17
`$s0`←2

23	
?	
2	
1	
3	
17	
2	
	←`$sp`

`$a0`←1
`$a1`←2
`$a3`←3
`$v0`←2
`$ra`←17
`$s0`←2

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←1
`$a1`←2
`$a3`←3
`$v0`←2
`$ra`←17
`$s0`←2

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←2
`$a1`←3
`$a3`←3
`$v0`←2
`$ra`←16
`$s0`←2

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←2
`$a1`←3
`$a3`←3
`$v0`←1
`$ra`←16
`$s0`←2

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←1
`$a1`←2
`$a3`←3
`$v0`←1
`$ra`←16
`$s0`←1

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←0
`$a1`←1
`$a3`←3
`$v0`←1
`$ra`←17
`$s0`←1

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	
17	
1	
	←`$sp`

`$a0`←0
`$a1`←1
`$a3`←3
`$v0`←1
`$ra`←17
`$s0`←1

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←1
`$a1`←1
`$a3`←3
`$v0`←1
`$ra`←17
`$s0`←1

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←1
`$a1`←2
`$a3`←3
`$v0`←1
`$ra`←17
`$s0`←1

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←0
`$a1`←2
`$a3`←1
`$v0`←1
`$ra`←18
`$s0`←1

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	
18	
1	
	←`$sp`

`$a0`←0
`$a1`←2
`$a3`←1
`$v0`←1
`$ra`←18
`$s0`←1

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←0
`$a1`←2
`$a3`←1
`$v0`←1
`$ra`←18
`$s0`←1

23	
?	
2	
1	
3	
17	
2	
1	
2	
3	←`$sp`

`$a0`←0
`$a1`←2
`$a3`←1
`$v0`←1
`$ra`←17
`$s0`←2

23	
?	
2	
1	
3	←`$sp`

`$a0`←0
`$a1`←2
`$a3`←1
`$v0`←1
`$ra`←17
`$s0`←2

23	
?	
2	
1	
3	←`$sp`

`$a0`←2
`$a1`←1
`$a3`←1
`$v0`←1
`$ra`←17
`$s0`←2

23	
?	
2	
1	
3	←`$sp`

`$a0`←1
`$a1`←1
`$a3`←2
`$v0`←1
`$ra`←18
`$s0`←2

23	
?	
2	
1	
3	
18	
2	
	←`$sp`

`$a0`←1
`$a1`←1
`$a3`←2
`$v0`←1
`$ra`←18
`$s0`←2

23	
?	
2	
1	
3	
18	
2	
1	
1	
2	←`$sp`

`$a0`←1
`$a1`←1
`$a3`←2
`$v0`←1
`$ra`←18
`$s0`←2

23	
?	
2	
1	
3	
18	
2	
1	
1	
2	←`$sp`

`$a0`←1
`$a1`←2
`$a3`←2
`$v0`←1
`$ra`←16
`$s0`←2

23	
?	
2	
1	
3	
18	
2	
1	
1	
2	←`$sp`

`$a0`←1
`$a1`←2
`$a3`←2
`$v0`←3
`$ra`←16
`$s0`←2

23	
?	
2	
1	
3	
18	
2	
1	
1	
2	←`$sp`

`$a0`←1
`$a1`←1
`$a3`←2
`$v0`←3
`$ra`←16
`$s0`←3

23	
?	
2	
1	
3	
18	
2	
1	
1	
2	
17	
3	
	←`$sp`

`$a0`←0
`$a1`←3
`$a3`←2
`$v0`←3
`$ra`←17
`$s0`←3

23	
?	
2	
1	
3	
18	
2	
1	
1	
2	←`$sp`

`$a0`←1
`$a1`←1
`$a3`←2
`$v0`←3
`$ra`←17
`$s0`←3

23	
?	
2	
1	
3	
18	
2	
1	
1	
2	
18	
3	
	←`$sp`

`$a0`←0
`$a1`←1
`$a3`←3
`$v0`←3
`$ra`←18
`$s0`←3

23	
?	
2	
1	
3	
18	
2	
1	
1	
2	←`$sp`

`$a0`←0
`$a1`←1
`$a3`←3
`$v0`←3
`$ra`←18
`$s0`←3

23	23	23	←$sp
?	?	?	
2	2	2	
1	1	1	
3	3 ←$sp	3 ←$sp	
18			
2			
1			
1			
2 ←$sp			

Première pile :
$a0←0
$a1←1
$a3←3
$v0←3
$ra←18
$s0←2

Deuxième pile :
$a0←0
$a1←1
$a3←3
$v0←3
$ra←18
$s0←2

Troisième pile :
$a0←0
$a1←1
$a3←3
$v0←3
$ra←23
$s0← ?

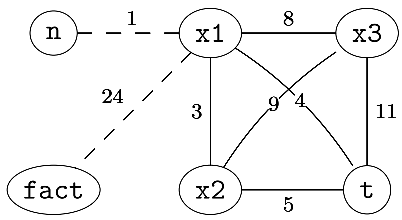
Quatrième pile :
$a0←0
$a1←1
$a3←3
$v0←3
$ra←23
$s0← ?

Exercice 5 : Allocation de registres (5 points)

1. Les prédécesseurs de l'instruction de la ligne 7 sont les instructions lignes 6 et 22.
   L'instruction précédant l'instruction de la ligne 23 est celle ligne 7.
2. Il était probablement plus judicieux d'utiliser l'algorithme de point fixe que la définition.

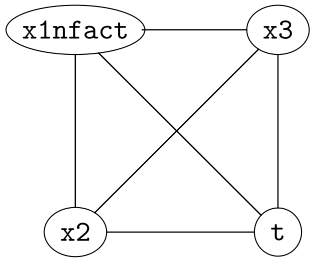
point	vivantes avant	vivantes après
1	n	x1
2	x1	x2
3	x2	x1 x2
4	x1 x2	x1 t
5	x1 t	x2 t
6	x2 t	x1 x2
7	x1 x2	x1 x2
8	x1 x2	x1 x3
9	x1 x3	x1 x2 x3
10	x1 x2 x3	x1 x2 t
11	x1 x2 t	x1 x3 t
12	x1 x3 t	x2 x3 t
13	x2 x3 t	x1 x2 x3
14	x1 x2 x3	x1 x3 t
15	x1 x3 t	x2 x3 t
16	x2 x3 t	x1 x2 x3
17	x1 x2 x3	x1 x3
18	x1 x3	x1 x2
19	x1 x2	x1 t
20	x1 t	x2 t
21	x2 t	x1 x2
22	x1 x2	x1 x2
23	x2	x1
24	x1	

3.

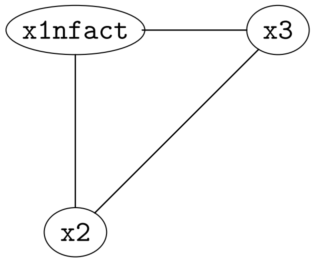


4. Aucun nœud n'est simplifiable (`n` et `fact` sont trivialement colorables mais possèdent une arête de préférence).

   On peut fusionner les arêtes de préférence en suivant le critère de George (le critère de Briggs ne marche pas ici).



   On ne peut pas simplifier (degré égal à trois pour chaque nœud, donc aucun trivialement colorable), ni fusionner ni geler (plus d'arêtes de préférence), on spille par exemple `t`.



   Chaque nœud est alors trivialement colorable, on peut simplifier `x1nfact`, `x2` et `x3`.

   En remontant les étapes, on donne la couleur 1 à `x1nfact`, la couleur 2 à `x2`, la couleur 3 à `x3`. On ne peut pas donner de couleur à `t` qui est donc effectivement spillé. Au final on obtient :