# Matching d’un nombre quelconque d’enfant dans une chaîne

**Solution par défaut :**

Axiom : I(5)[I(4)…] …

production:

I(p) > [I(p1)] [I(p2)] [I(p3)]I(pa)

I(p) > [I(p1)] [I(p2)] [I(p3)]

I(p) > [I(p1)] [I(p2)] I(pa)

I(p) > [I(p1)] [I(p2)]

I(p) > [I(p1)] I(pa)

I(p) > [I(p1)]

I(p) > I(pa)

I(p)

Solution fastidieuse (et limitée par rapport aux nombres d’enfants).

**Utilisation du module x :**

Pour le matching des modules I lateraux, on peut utiliser les expressions régulières de lpy, notamment le module x.

Définition : x(pattern [,nb occurrence]) ou x(pattern [,nb occurrence min, nb occurrence max]). Par défaut, le nombre min et max d’occurrence sont égal respectivement à 0 et l’infini.

Par défaut, le motif sera matché 0,1 ou plusieurs fois et dans ce cas les paramètres seront stockés sous forme de liste.

Exemple :

Axiom : I(0) I(1) I(2) I(3) I(4)

production :

x(I(p)) :

print p

>>> [0,1,2,3,4]

On peut alors redéfinir les règles précédentes de la manière suivante

I(p) > x([I(pl)]) I(pa)

I(p) > x([I(pl)])

**Utilisation du module or :**

Pour simplifier, les 2 règles précédentes en une seule, on peut également utiliser le module or des expressions régulières de lpy.

Ce module permet de matcher de manière alternative 2 motifs.

Définition : or(pattern1 , pattern2, …)

Le motif matché aura ses valeurs de paramètres mises à jour. Pour les autres les valeurs None seront utilisées.

Par exemple

Axiom : A(3)

Production :

or(A(p),B(q)) :

print p,q

>>> 3, None

On va également utiliser le module None qui est toujours matché.

Axiom : A(3)B(5) :

Production :

A(p) None B(q) :

print p,q

>>> 3,5

Du coup les règles peuvent du matching peuvent se réécrire de la manière suivante :

I(p) > x([I(pl)]) or(I(pa),None)

**Solution alternative en utilisant la commande InRightContext**

Les versions les plus récentes de lpy, on une fonction InRightContext qui permet de rechercher un contexte directement dans le code de la règle et d’utiliser les commande if et while pour moduler la recherche de contexte

InRightContext(pattern, paramvalue)

Dans ce cas la récupération des paramètres est un peu plus compliquée mais on a la souplesse du langage pour créer son contexte.

Dans notre cas, la recherche du contexte se fera de la manière suivante :

I(v) :

p,a = [], None

params = {}

while InRightContext([I(p)], params):

p.append(params['p'])

if InRightContext(I(a), params):

a = params['a']

print v, p,a