**Documentation du système de règles d’analyse syntaxique**

**Initialisation**

***Succession entre arbres et succession immédiate entre tokens***

Le module *suc\_init* initialise une relation ***SUC*** qui représentera la relation de succession immédiate entre arbres partiels de dépendance via leurs racines, tandis que la relation « < » marquera la relation de succession immédiate entre tokens.

Au départ, les arbres se réduisant aux tokens, la relation ***SUC*** est introduite entre deux tokens qui se suivent immédiatement.

***Parties du discours***

On s’intéresse aux mots de la phrase pris isolément. A partir de leur trait *pos*, qui définit leur partie du discours raffinée par des informations morphologiques, on leur ajoute des traits morphologiques à l’aide du module *pos\_unfold*.

***Particularités de Sequoia***

Puis des modules sont dédiés au traitement de particularités de l’annotation du corpus Sequoia :

* les expressions étrangères qui, dans le module *foreign-expr\_compos*, seront considérés comme un tout dont la tête est le dernier mot ;
* les expressions qui sont considérées comme mots composés, et donc comme tokens uniques, mais qui ne devraient pas l’être ; elles sont décomposées à l’aide du module *word\_compos ;*
* le dépliage des contractions phonologiques à l’aide du module *phon\_unfold*;
* la correction de traits erronés à l’aide du module *feat\_correct.*

***Mots composés et expressions idiomatiques***

Puis les préfixes sont rattachés au mot qui suit dans le module *prefix* et, dans le module *frozen\_expr\_compose*, on détermine les expressions figées qu’on traite de deux façons :

* Certaines sont considérées comme des mots uniques composés. Leur tête est rattachée à ses dépendants par des dépendances ***mwe****.* Cela permet de les distinguer des dépendances externes mais à la fin de l’analyse, elles sont ré-étiquetées de façon standard. Ces expressions sont listées dans des lexiques. Ce sont des expressions de la forme *adjectif =* *préposition + nom*, *adjectif =* *préposition + déterminant + nom, adverbe = adverbe + mot, adverbe = conjonction de subordination + mot, adverbe = interjection + adverbe, adverbe = préposition + mot, adverbe = préposition + déterminant + nom, conjonction de subordination = conjonction de subordination + adverbe, conjonction de subordination = préposition + nom + conjonction de subordination, conjonction de subordination = verbe, nom = préposition + nom, numéral = numéral + préposition + numéral, préposition = préposition + déterminant + nom + préposition, préposition = préposition + nom + préposition,* ou encore *préposition = clitique + verbe + « de »*.
* Pour d’autres, on conserve la structure syntaxique standard. Ce sont les couples de cardinaux reliés par une préposition particulière, la préposition « près de » suivie d’un cardinal et enfin de certains adverbes suivis de certains mots.

***Ajout de traits***

Ensuite, on introduit des traits supplémentaires nécessaire à l’analyse syntaxique à l’aide du module *feature\_add*. Un trait *type* est ajouté à certaines prépositions listées dans le lexique *typed\_prep.lp* avec les valeurs suivantes : *deloc* (lieu d’origine), *loc* (locatif), *temp* (temporel).

Le trait type est aussi affecté à certains noms communs listés dans le lexique *typed\_noun.lp* avec les valeurs suivantes :

* *abs* : noms abstraits (« désir »),
* *loc*: noms exprimant un complément locatif (« place »),
* *temp* : noms exprimant un complément temporel (« jour »).

Les adverbes sont rangés dans différentes classes à l’aide des traits à valeur booléenne suivants:

* *adv\_adj*: adverbe se comportant comme un adjectif (« bien »),
* *adv\_adjmod*: modificateur d’adjectif (« cliniquement »),
* *adv\_amod* : modificateur d’adjectif ou d’adverbe (« très »),
* *adv\_loc*: locatif (« ici » ),
* *adv\_man* : de manière (« fermement »),
* *adv\_nmod* : modificateur de nom (« à la fois »),
* *adv\_pmod* : modificateur de préposition (« à la fois »),
* *adv\_qmod* : modificateur d’expression quantifiée (« presque »),
* *adv\_quant* : exprimant une quantification (« beaucoup »),
* *adv\_smod* : modificateur de phrase (« alors »),
* *adv\_temp*: temporel (« aujourd’hui »),
* *adv\_vmod* : modificateur de verbe (« vite »)

Ces adverbes sont listés dans des lexiques correspondant à ces différents traits. Pour les adverbes non présents dans ces lexiques, le module *deflt\_feature\_add* ajoute le trait *adv\_man = true*.

Pour les adverbes munis de l’un des traits *adv\_loc* *= true*, *adv\_man*  *= true*, *adv\_temp* *= true*, est ajouté systématiquement le trait *adv\_compl = true*, qui permet de reconnaître les adverbes compléments.

***Coordination lexicale***

Le module *lex\_coord* réalise la coordination lexicale des adverbes, adjectifs, déterminants, noms communs et prépositions. Il le fait de façon grossière dans le cas où deux mots de même catégorie sont séparés par une conjonction de coordination. Cela peut entraîner quelques erreurs, notamment pour les noms communs. Il traite aussi de façon récursive des coordinations lexicales de plus de deux mots.

***Limites des phases, propositions et citations***

Certaines citations sont encadrées par des guillemets anglais. Le module *french\_quotation* permet de distinguer les guillemets ouvrant des guillemets fermant et de les remplacer par des guillemets français. Le module *dashes* traite les expressions entre tirets comme des expressions entre parenthèses, si bien qu’il les remplace par des parenthèses.

Le module *clause\_bound* est ensuite consacré au marquage des débuts de phrases et de propositions à l’aide des traits respectifs *sent\_start=y* et *cl\_start=y*. Il ajoute aussi un signe de fin de phrase particulier quand il n’y en a pas.

Le trait *sent\_start=y* est ajouté aux tokens suivants :

* Début de l’énoncé,
* Signe de ponctuation juste après le début de l’énoncé,
* Signe de ponctuation marquant la fin d’une phrase suivie d’un token,
* Parenthèse, crochet ou guillemet français ouvrant.

Le trait *cl\_start=y*, outre dans les cas précédents, est ajouté aux tokens suivants :

* Les conjonctions dont on est sûr d’après un lexique qu’elles introduisent une proposition,
* Pour les subordonnées relatives ou interrogatives introduites par un mot relatif ou interrogatif indirect, ce mot,
* Pour les subordonnées relatives ou interrogatives introduites par un mot relatif ou interrogatif direct, le token précédant ce mot,
* Les pauses quand elles suivent immédiatement un token marqué comme début de proposition.

**Construction des arbres élémentaires**

On construit les arbres de dépendance des noyaux des syntagmes nominaux, verbaux, adjectivaux et adverbiaux.

Dans la suite des modules, on retrouvera périodiquement le module *quotes* qui traite des expressions entre guillemets : lorsqu’une telle expression est totalement analysée, les guillemets sont ajoutés à son arbre de dépendance à l’aide de deux dépendances ***ponct***.

***Détermination de certains modificateurs étroitement liés à leur gouverneur***

Dans le module *adv\_modif,* on cherche à déterminer un certain nombre de modificateurs d’adverbes, en général d’autres adverbes. Pour ce qui concerne les modificateurs à gauche :

* le déterminant « le » modifiant l’adverbe « plus » ou « moins » pour en faire un adverbe superlatif, ce qui exclut l’adjectif au superlatif (« la plus grande ville »),
* les adverbes avec le trait *adv\_amod=true* qui ne peuvent pas modifier les racines d’arbres juste à gauche,
* les adverbes avec le trait *adv\_amod=true* qui peuvent aussi modifier des verbes finis simples ou au participe présent situés juste avant, dans un contexte où de tels verbes ne sont pas présents,
* les adverbes modifiant d’autres adverbes placés juste après selon un lexique donné *adv\_mod\_adv.lp*.

Pour ce qui concerne les modificateurs à droite d’adverbes, dans le même module, deux règles leur sont dédiés, selon que le modificateur est suivi ou non par un signe de ponctuation.

Le module *conj-prep\_modif*, ensuite, traite des adverbes modifiant les conjonctions ou prépositions placées juste après et le module *adj\_modif* traite des adverbes modifiant des adjectifs placés juste après. Enfin, le module *pro\_modif* traite des adverbes modifiant les pronoms placés juste avant.

Après, le module *adj\_modif* est dédié aux modificateurs d’adjectifs placés avant ceux-ci. On distingue les cas suivants :

* le modificateur est un adverbe avec le trait *adv\_adjmod=true* ou *adv\_amod=true* qui ne peut pas modifier la racine de l’arbre qui le précède immédiatement ;
* le modificateur est un adverbe avec le trait *adv\_adjmod=true* ou *adv\_amod=true* qui peut modifier un verbe fini ou un participe présent, mais la racine de l’arbre qui le précède immédiatement n’en est pas une ;
* le modificateur est l’adverbe « quelque » et il est suivi immédiatement par un numéral.

Ensuite, les modules *det\_modif* et *pro\_modif* traitent respectivement des modificateurs de déterminants et de pronoms.

***Construction du noyau verbal***

Dans les modules *verb\_left-clitic* et *verb\_right-clitic*, on introduit les dépendances entre clitiques et leur verbe gouverneur en se fondant uniquement sur la morphologie des clitiques et en gardant ambiguë la dépendance quand la morphologie ne permet pas de trancher. Le clitique « en » est rattaché au verbe qu’il modifie même lorsqu’il dépend de son objet *(« Jean en mange la moitié »*).

Les modules *verb\_left-modif* et *verb\_right-modif* visent à réaliser les dépendances entre les verbes et certains adverbes et pronoms adjacents au noyau dont ils sont la tête. On distingue les deux modules pour le cas des verbes composés où un modificateur se situant entre l’auxiliaire et le participe passé est systématiquement rattaché au participe passé.

Le module *verb\_aux* réalise les dépendances pour les verbes composés entre le verbe et son auxiliaire de temps, passif ou causatif. A la différence de l’annotation de Sequoia, il considère aussi les verbes modaux « devoir » et « pouvoir » comme des auxiliaires.

***Construction du noyau nominal***

La construction commence par la réalisation des dépendances des adjectifs épithètes à gauche à l’aide du module *noun\_left\_adj*.

Ensuite, ce sont les dépendances vers les déterminants à l’aide du module *noun\_det*. Le déterminant interrogatif « quel » est traité de façon spécifique car nous avons besoin d’indiquer que le syntagme nominal auquel il contribue est une *expression interrogative*. Une *expression interrogative (relative)* est une expression contenant un mot interrogatif (pronom relatif).

Enfin, le module *noun\_mod* réalise les dépendances des modificateurs périphériques du noyau nominal : certains adjectifs ou adverbes situés à gauche du déterminant et certains noms à la droite du nom de tête.

**Détermination des arguments des prépositions et des adverbes ainsi que ceux des verbes et des adjectifs qui leur sont liés étroitement en étant placés après**

Les arguments du verbe concernés sont les arguments directs placés juste après le verbe : les sujets inversés, les attributs du sujet ou de l’objet et les objets directs. Cette détermination s’entrecroise avec la construction du noyau prépositionnel, étant la dépendance étroite entre les deux processus.

***Arguments des prépositions et des adverbes qui ne sont pas des propositions***

Le module *prep\_obj* réalise les dépendances des prépositions vers leur argument, lorsque ce dernier est un nom, un pronom, un adjectif ou un adverbe. Il ne prend pas en compte les prépositions qui ont comme argument un verbe car il faut d’abord réaliser les arguments étroits du verbe. Chaque règle correspond à un cas particulier :

* le cas standard,
* le cas où certaines prépositions (la préposition « sauf ») ont comme argument une autre préposition,
* le cas où l’argument est une expression interrogative ou relative.

Toujours dans le module *prep\_obj*, la règle *avec\_mod* traite des modificateurs de la préposition « avec » tels que dans le syntagme prépositionnel « avec pour couleur de fond le rouge ». Enfin, des règles assurent le typage des prépositions selon les valeurs *deloc*, *loc* et *temp*.

Le module *adv\_compl* réalise les compléments requis des adverbes.

***Arguments des verbes qui leur sont liés étroitement et qui sont placés après***

Les sujets inversés sont ensuite réalisés par le module *rev\_subject*. Ce module est placé après *prep\_obj* pour les sujets inversés de verbes qui se situent juste après un syntagme prépositionnel relatif ou interrogatif (*« avec qui vient Jean »*). En effet, il faut d’abord réaliser le syntagme prépositionnel pour détecter l’inversion du sujet. On distingue différents cas de sujets inversés :

* un verbe est immédiatement précédé par une expression relative ou interrogative et il est immédiatement suivi par un groupe nominal ;
* un verbe est en tête d’une proposition (repérée par le trait *cl\_start=y*) et il est immédiatement suivi par un groupe nominal ;
* un verbe est immédiatement précédé par une expression qui est potentiellement un modificateur placé en tête de proposition et il est immédiatement suivi par un groupe nominal ; éventuellement, le verbe est séparé de son modificateur par une virgule ;
* le cas particulier de « tel » suivi d’un verbe fini et après d’un groupe nominal.

Après les sujets inversés, on traite spécifiquement les locutions verbales à l’aide des modules *aff\_verbal\_locution* et *verbal\_locution*. Elles sont identifiées à l’aide de lexiques qui se différencient les uns des autres par le type de complément qui suit le verbe tête de la locution. Ces locutions sont considérées comme formant un verbe composé, ce qui est exprimé par une dépendance ***mwe*** entre le verbe tête et son dépendant.

On traite ensuite les compléments directs de verbe, en commençant par ceux qui se rapportent à des verbes prenant un pronom clitique à valeur non argumentale et un complément direct non verbal à l’aide du module *aff\_verb\_direct\_comp*. Ensuite, le module *verb\_direct\_comp* réalise les compléments directs non verbaux des autres verbes. Puis, on reprend la même distinction mais pour les compléments directs qui sont des infinitifs avec les modules *aff\_verb\_direct\_inf* et *verb\_direct\_inf*. Cette séparation en deux étapes, compléments non verbaux puis compléments verbaux, se justifie par les configurations *verbe + infinitif ou participe présent + complément direct*, où il est plus simple de commencer par rattacher le complément direct à l’infinitif ou au participe présent.

***Arguments directs sous forme de propositions des prépositions et des adjectifs***

Le module *prep\_verb* réalise les dépendances des prépositions vers leur objet lorsque celui-ci est un participe ou un infinitif. Ce module se situe après *aff\_verb\_direct\_inf* et *verb\_direct\_inf* car une préposition peut gouverner un participe ou un verbe à l’infinitif qui a un complément direct ; il vaut alors mieux rattacher le complément avant la préposition.

Le module *adj\_inf* est dédié aux adjectifs qui ont un infinitif comme attribut du sujet (« censé », « réputé »).

**Sujets canoniques des verbes**

Cette phase inaugure la partie itérative de la séquence.

Le module *can\_subject* détermine les sujets des verbes quand ils précèdent ceux-ci. Ce module est dans la partie itérative de la séquence car un sujet canonique peut être séparé du verbe auquel il se rattache par un nombre indéterminé de mots, qu’il faut rattacher d’abord.

La condition commune d’application des règles du module est que le verbe concerné V soit précédé par une groupe nominal. On note la tête du groupe nominal N et on distingue plusieurs cas :

* N est un pronom relatif sujet « qui » ou « lequel » ;
* l’arbre de N se situe au début d’une proposition ;
* l’arbre de N est immédiatement précédé par un arbre représentant un modificateur possible du verbe (les deux arbres peuvent séparés par une virgule).

Le module prend aussi en compte le cas des infinitifs sujets.

**Attachement des conjonctions et des mots interrogatifs et relatifs**

Le module *cs\_dep* introduit une dépendance ***obj.cpl*** d’une conjonction de subordination à la racine d’un arbre, lorsque celle-ci est la tête d’une proposition et lorsque la couverture de l’arbre[[1]](#footnote-1) suit immédiatement la conjonction.

Le module *wh\_dep* introduit une dépendance qui a pour cible une expression interrogative ou relative et pour source le verbe tête de la proposition qui suit. Cette dépendance est provisoire et elle sera remplacée dans les modules finaux de mise à jour des fonctions syntaxiques par une dépendance qui indique la fonction syntaxique effective de l’expression en question.

L’étiquette attachée à cette dépendance provisoire dépend de la forme de l’expression interrogative ou relative d’où différents cas qui sont traités par différentes règles :

* Arguments directs sauf dans le cas du pronom *« qui »*, qui peut être aussi sujet ; c’est l’étiquette ***ATS-OBJ*** qui est attachée.
* Pronom interrogatif *« qui »* attribut ou objet direct ; c’est aussi l’étiquette ***ATS-OBJ*** qui est attachée. On a la garantie de cette fonction du pronom *« qui »* parce que le verbe a déjà un sujet et les prépositions sont déjà rattachées à leur objet.
* Compléments indirects ; c’est l’étiquette ***mod*** qui est attachée.

Le module *cc\_dep* introduit une dépendance ***dep.coord*** d’une conjonction de coordination à la racine de l’arbre qui suit immédiatement la conjonction. Les conjonctions corrélées *soit…soit* et *ni…ni* sont traitées différemment. Elles sont rattachées par une dépendance ***dep*** à la racine de l’arbre qui suit chacune d’elles.

**Extension des arbres de dépendances à droite**

Le processus est réalisé par le module *right\_extend\_compl*.

***Principe***

On considère une phrase partiellement analysée en dépendances, où tous les arbres partiels déjà construits ont leur couverture qui forme un segment continu.

On cherche les arbres partiels AD, appelés *arbres à extension droite*, tels que tout arbre partiel AE qui suit immédiatement AD, dont la racine RE fait partie d’une des catégories listées ci-après a sa racine RE qui dépend d’un mot de la frontière droite de AD [[2]](#footnote-2).

A partir d’un arbre à extension droite AD, on considère l’arbre AE qui suit immédiatement l’arbre AD. On essaie d’attacher par une dépendance sa racine RE à un nœud N de la frontière droite de AD, N étant le gouverneur de RE.

On itère le processus aussi longtemps que possible, c’est-à-dire tant que RE est dans une des catégories suivantes :

* ADJ : adjectif ou mot pouvant se comporter comme adjectif (participe ou nom sans déterminant mais non précédé d’une virgule),
* ADV : adverbe avec le trait *adv\_compl=true,*
* APP : nom ou mot se comportant comme un nom (adjectif ou participe tête d’un groupe nominal) précédé d’une virgule, d’un tiret, d’une parenthèse ouvrante ou de deux points, se comportant potentiellement comme une apposition,
* COORD : conjonction de coordination gouvernant la tête d’un second conjoint potentiel,
* CS : conjonction de subordination différente de « que » introduisant potentiellement une subordonnée se comportant comme un modificateur,
* INF : infinitif se comportant potentiellement comme le complément direct d’un verbe,
* INTER : verbe tête d’une proposition interrogative indirecte,
* NOUN : nom déterminé, pronom, participe ou adjectif déterminés se comportant comme compléments directs d’un verbe,
* PARV : verbe fini qui a une virgule comme dépendant et qui est potentiellement la tête d’une proposition incise,
* PREPN : préposition gouvernant un nom, un adjectif ou un participe passé,
* PREPV : préposition gouvernant un verbe à l’infinitif ou au participe présent,
* QUE : conjonction « que » introduisant une subordonnée qui peut dépendre d’un verbe ou d’un nom,
* RELV : un verbe tête d’une proposition relative précédée ou non par une virgule ou un tiret.

***Détermination des racines d’arbres à extension droite***

Soit un arbre AD que l’on cherche à caractériser comme arbre à extension droite. Il est immédiatement précédé par PRE qui peut être un arbre suivi d’une pause (virgule ou tiret) ou ou un adverbe, ou encore qui peut être vide. PRE est immédiatement précédé par START qui peut être marqué comme début de proposition ou une pause. AD est immédiatement suivi par un arbre AE.

Selon la nature de START, on distingue deux cas :

* START est marqué comme début d’une proposition. Ce cas se divise ensuite deux selon la nature de PRE :
	+ Si PRE est vide ou un arbre suivi d’une pause, selon que AD représente une expression qui pour tête une conjonction ou pas, on applique un traitement différent.
	+ Si PRE est un adverbe, AD ne peut pas avoir comme tête une conjonction.
* START est une pause qui n’est pas marquée comme début d’une proposition et AD représente une proposition relative.

Dans tous les cas, on pose comme condition supplémentaire que AD puisse s’étendre à droite. On vérifie que AE, l’arbre qui suit immédiatement AD, a une racine qui possède l’une des trois propriétés suivantes :

* C’est un adjectif, un nom, une préposition, un pronom, un infinitif ou un participe.
* C’est une conjonction tête de toute une expression.
* C’est la tête d’une proposition relative, interrogative ou incise.

***Les probabilités à la rescousse pour la détermination du gouverneur d’un dépendant à droite***

Soit un arbre à extension droite AD de racine RD et un arbre AE de racine RD immédiatement à droite de AD candidat à à une extension de AD. Compte tenu de la propriété de AD, le gouverneur de RE est un élément de la frontière droite de AD. La catégorie grammaticale, les propriétés morphosyntaxiques de RD et de RE sont le premier élément qui permet de déterminer le gouverneur de RE mais il reste une ambiguïté qui est levée en utilisant des probabilités calculées comme statistiques sur un corpus.

On calcule sur un corpus de référence annoté en dépendances syntaxiques le nombre de dépendances d’un mot w1 vers un mot w2 donnés dans les cas suivants :

* w2 est un adjectif et w1 est un verbe ou un nom gouverneur de w1,
* w2 est un nom en apposition par rapport à w1 qui est aussi un nom,
* w2 et w1 sont les têtes de deux conjoints coordonnés,
* w2 est une conjonction de subordination et w1 est un adjectif, un nom ou un verbe gouverneur de w2,
* w2 est un nom dépendant via une préposition de w1 qui est un adjectif, un nom ou un verbe
* w2 est un infinitif dépendant via une préposition de w1 qui est un adjectif, un nom ou un verbe

L’ensemble de ces calculs sont conservés dans ce que nous appelerons par la suite le *registre des fréquences*. RE est pourvu d’un trait *freq* qui donne la fréquence estimée de la dépendance qui le lie à un gouverneur provisoire choisi sur la frontière droite de AD. Cette fréquence est estimée à partir du registre des fréquences.

***Mise en œuvre de l’extension à droite des arbres de dépendance***

Soit un arbre partiel à extension droite AD, qui est suivi immédiatement par un arbre partiel AE de racine RE candidat à l’extension de AD

La frontière droite de AD est parcourue de la racine RD vers les feuilles à la recherche du point d’ancrage de la source d’une dépendance qui a pour cible RE. On utilise pour cela trois dépendances provisoires :

* ROOT.PART de RD vers RE,
* GOV.PROV du nœud de la frontière droite de AD considéré provisoirement comme gouverneur de RE vers RE,
* GOV.TEST du nœud de la frontière droite de AD testé comme gouverneur de RE vers RE.

Décrivons maintenant l’algorithme qui permet de réaliser l’attachement de AE à AD.

***Entrée****: un arbre de dépendance à extension droite AD de racine RD et un arbre AE de racine RE situé immédiatement à sa droite.*

***Sortie****: une dépendance d’un nœud de la frontière droite de AD vers RE*

***Algorithme****:*

1. *Initialiser les dépendances ROOT.PART, GOV.PROV et GOV.TEST comme trois dépendances de RD vers RE.*
2. *Initialiser le trait* freq *de RE à 0.*
3. ***Faire****:*
	1. *Tester la compatibilité de la source TEST de GOV.TEST avec RE à l’aide de règles spécifiques à chacune des catégories possibles de RE listées plus haut.*
	2. ***Si*** *TEST est un gouverneur possible de RD* ***alors****:*
		1. *Mettre la valeur du trait* freq *dans* freq\_old *et mettre dans* freq\_new *la valeur qui est celle donnée par le registre des fréquences, si elle si trouve, sinon elle est mise à 1.*
		2. *Comparer les valeurs de* freq\_old *et* freq\_new *et en fonction de la comparaison mettre à jour GOV et* freq*.*
	3. ***Si*** *TEST a un fils sur la frontière droite de RD* ***alors****:*
		1. *Remplacer TEST par ce fils*

***Sinon***

* + 1. *Supprimer les dépendances ROOT.PART et GOV.TEST et remplacer GOV.PROV par la dépendance adéquate.*

***Jusqu’à*** *suppression de ROOT.PART, GOV.PROV et GOV.TEST*

Pour un arbre AD donné, cet algorithme est appliqué tant qu’il existe un arbre AE candidat à son extension droite. Le processus d’extension s’arrête lorsque AE se réduit à un signe de ponctuation, un verbe fini qui n’est pas la tête d’une relative ou d’une proposition incise, une expression interrogative ou relative ou à une conjonction isolée.

**Expression en début de proposition se rattachant à droite à la tête sans en être le sujet**

On considère une proposition dont le début est le token START marqué comme tel. Juste après START, nous avons un arbre T1 de racine R1 qui n’est pas un verbe fini. T1 est immédiatement suivi par un arbre T2 dont la racine R2 est la tête de la proposition. Entre les deux, il y a éventuellement une virgule PAUSE1. R1 est alors rattachée à R2 par les règles du module *left\_compose*.

On considère deux cas selon la tête de la proposition.

***La tête R2 de la proposition est un verbe fini***

Dans ce cas on subdivise le traitement en deux selon l’existence ou non de PAUSE1 :

* Si PAUSE1 existe, on distingue 6 traitements selon la nature de R1 :
	+ modificateur avec les règles *mod\_comma\_fin-verb1* et *mod\_comma\_fin-verb2* (quand le modificateur est une proposition subordonnée ou un syntagme prépositionnel, on doit vérifier que la conjonction ou la préposition correspondante a un dépendant),
	+ modificateur précédé d’une virgule, avec la règle *comma\_mod\_comma\_fin-verb1* et *comma\_mod\_comma\_fin-verb2*,
	+ nom adverbial avec la règle *circ-noun\_comma\_fin-verb*,
	+ nom ou pronom exprimant une dislocation à gauche avec la règle *dis-cat\_comma\_fin-verb* qui établit la dépendance ***dis*** de R2 vers R1,
	+ nom référant au sujet de la proposition se comportant comme un modificateur du verbe principal avec la règle *mod-noun\_comma\_fin-verb* qui établit une dépendance ***mod*** de R2 vers R1,
	+ vocatif avec la règle *voc-noun\_comma\_fin-verb* qui établit une dépendance ***mod.voc*** de R2 vers R1.
* Si PAUSE1 n’existe pas, on distingue deux cas selon la nature de START :
	+ START est marqué comme début d’une proposition et R1 est un modificateur potentiel de R2. C’est la règle *mod\_fin-verb1* ou *mod\_fin-verb2* qui intervient (quand le modificateur est une proposition subordonnée ou un syntagme prépositionnel, on doit vérifier que la conjonction ou la préposition correspondante a un dépendant).
	+ START est une virgule qui n’est pas marquée comme début de proposition et R1 est une proposition ou un adverbe. C’est la règle *comma\_mod\_fin-verb* qui intervient.

***La tête R2 de la proposition est un nom***

Alors il est immédiatement suivi par une marque de fin de phrase. On distingue alors deux cas selon l’existence dePAUSE1 :

* Si PAUSE1 existe, R1 est un modificateur de R2. C’est la règle *mod\_comma\_noun\_stop* qui intervient.
* Si PAUSE1 n’existe pas, R1 est un adverbe aussi modificateur de R2. C’est la règle *adv\_noun\_stop* qui intervient.

**Expression isolée par la ponctuation se rattachant à un gouverneur à gauche**

Le module *right\_insertion* traite des expressions entre virgules, parenthèses ou tirets (dans la suite de la section, nous utiliserons le mot « pause » pour exprimer indifféremment une virgule, une parenthèse ou un tiret), qui se rattachent à un gouverneur à gauche. La pause à droite de l’expression peut être remplacée par une fin de phrase.

On appelle T1 l’arbre à gauche auquel l’arbre T2 entre pauses doit être rattaché. Les racines de T1 et T2 sont désignées par R1 et R2. Les deux signes de ponctuation qui encadrent T2 sont notés de gauche à droite PAUSE1 et PAUSE2.

***L’expression insérée n’est pas une proposition (R2 n’est pas un verbe fini)***

On distingue deux cas selon la nature de R1 :

* R1 n’est pas un verbe fini. Dans ce cas, on considère l’arbre qui suit immédiatement PAUSE2.
	+ Si sa racine POST n’est pas un verbe fini, on considère la nature de R1.
		- Si R1 n’est pas un adverbe, on considère la nature de R2.
			* si R2 n’est pas un mot qui a plus de chances de se rattacher à droite au verbe principal de la proposition (adverbe, préposition ou conjonction de subordination), on fait l’hypothèse qu’il se rattachera à la tête de la proposition ou à l’arbre T1. La règle *initcl\_cat\_pause\_right-mod\_pause* a pour effet de rattacher la pause PAUSE1 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2 et POST.
			* Si R2 est un mot qui a plus de chances de se rattacher à droite au verbe principal de la proposition (adverbe, préposition ou conjonction de subordination), la règle *initcl\_cat\_pause\_left-mod\_pause* va rattacher provisoirement R2 et PAUSE1 à R1 respectivement par une dépendance ***DEP.PROV*** et une dépendance ***ponct***
		- Si R1 est un adverbe, on fait l’hypothèse qu’il se rattachera à R2 ou à la tête de la proposition. De façon provisoire, la règle *initcl\_adv\_pause\_right-mod\_pause* va rattacher R1 et PAUSE1 à R2 respectivement par une dépendance ***DEP.PROV*** et une dépendance ***ponct***.
	+ Si la racine POST est un verbe fini, on considère deux cas selon qu’il a son sujet ou pas.
		- Si POST a son sujet, on regarde si R2 va plutôt dépendre de R1, auquel cas on applique la règle *initcl\_cat\_pause\_right-mod\_pause\_fin-verb1*. Celle-ci va rattacher le signe de ponctuation PAUSE1 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2. Si R2 va plutôt dépendre de POST, on applique la règle *initcl\_cat\_pause\_left-mod\_pause\_fin-verb1*. Celle-ci va rattacher le signe de ponctuation PAUSE2 et R2 à POST, respectivement par des dépendances ***ponct*** et ***mod***.
		- Si POST n’a pas son sujet, on fait l’hypothèse que R1 peut être ce sujet. On regarde ensuit si R2 va plutôt dépendre de R1, auquel cas on applique la règle *initcl\_cat\_pause\_right-mod\_pause\_fin-verb2*. Celle-ci va rattacher les signes de ponctuation PAUSE1 et PAUSE2 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2. Si R2 va plutôt dépendre de POST, on applique la règle *initcl\_cat\_pause\_left-mod\_pause\_fin-verb2*. Celle-ci va rattacher les signes de ponctuation PAUSE1 et PAUSE2 à POST par des dépendances ***ponct***, et R2 aussi à POST par une dépendance ***mod***.
* R1 est un verbe fini. On suppose que le début de la proposition est aussi un début de phrase et on considère alors PAUSE2.
	+ Si PAUSE2 est une vraie pause (virgule, parenthèse ou tiret), on considère la nature de R2.
		- Si R2 n’est pas une conjonction ni une préposition ou est un adverbe modificateur de phrase (avec le trait *adv\_smod=true*), on considère que T2 va venir étendre à droite T1. Les règles *initsent\_fin-verb\_pause\_mod\_pause* et *initsent\_fin-verb\_pause\_adv\_pause* rattacheront PAUSE1 et PAUSE2 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2.
		- Si R2 est une conjonction ou une préposition, elle doit être la tête d’une expression. On considère alors que T2 va venir étendre à droite T1. La règle *initsent\_fin-verb\_pause\_conj-prep\_pause* rattachera PAUSE1 et PAUSE2 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2.
	+ Si PAUSE2 est une fin de phrase, la règle *initsent\_fin-verb\_pause\_mod\_stop* rattachera PAUSE1 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2.

***L’expression insérée est une proposition relative***

On distingue alors deux cas selon la nature de PAUSE2.

* PAUSE2 est une vraie pause. On considère que T2 va venir étendre à droite T1. La règle *initcl\_cat\_pause\_rel-verb\_pause* rattachera PAUSE1 et PAUSE2 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2.
* PAUSE2 marque la fin d’une phrase. T2 va aussi venir étendre à droite T1, mais la règle *initcl\_cat\_pause\_rel-verb\_stop* rattachera seulement PAUSE1à R2.

***L’expression insérée est une proposition juxtaposée ou incise***

Le début de la proposition est aussi un début de phrase. R2 est un verbe fini et on distingue plusieurs cas selon la présence ou non d’un sujet de R2 situé avant et selon la nature de la ponctuation PAUSE1.

* PAUSE1 n’est pas un tiret, R2 a un sujet avant. On fait alors l’hypothèse que R1 et R2 sont simplement juxtaposées et on considère deux cas selon R2.
	+ R2 n’a pas de gouverneur. PAUSE1 est un signe de ponctuation quelconque à l’exception d’un tiret, une parenthèse ou un guillemet ouvrant. Nous sommes dans le cas d’une juxtaposition simple et la règle *initsent\_fin-verb\_punct\_fin-verb\_punct* rattache R2 à R1 par une dépendance ***mod*** et PAUSE1 aussi à R1 par une dépendance ***ponct***.
	+ R2 est gouverné par une conjonction de coordination et PAUSE1 est une virgule. On est dans le cas d’une juxtaposition et la règle *initsent\_fin-verb\_comma\_cc\_fin-verb\_ponct* rattache PAUSE1 à la conjonction de coordination, ce qui permettra ensuite d’étendre T1 avec l’arbre de racine la conjonction.
* R2 n’a pas de sujet avant mais un argument direct placé après. On fait l’hypothèse que cet argument est le sujet de R2 et que T2 représente une proposition incise. On considère deux cas selon la nature de PAUSE2.
	+ PAUSE2 est une vraie pause. Alors, la règle *initsent\_fin-verb\_pause\_fin-verb\_pause* rattachera PAUSE1 et PAUSE2 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2.
	+ PAUSE2 marque la fin d’une phrase. Alors, la règle *initsent\_fin-verb\_pause\_fin-verb\_stop* rattachera PAUSE1 à R2, ce qui permettra ensuite d’étendre à droite l’arbre T1 avec l’arbre T2.
* PAUSE1 est un tiret. On fait alors l’hypothèse que T2 représente une proposition incise. Selon la nature de PAUSE2, tiret ou fin de phrase, on applique la règle *initsent\_fin-verb\_dash\_fin-verb\_dash* ou *initsent\_fin-verb\_dash\_fin-verb\_stop*. Dans les deux cas, PAUSE1 est rattaché à R2, ce qui permettra ensuite d’étendre T2 avec T1. PAUSE2 est rattaché à R2 seulement dans le premier cas.

***Insertions particulières***

Le même module *right\_insert* traite d’insertions particulières :

* Lorsqu’une préposition ou une conjonction de coordination est suivie par un adverbe ou un groupe prépositionnel entre virgules, on considère qu’il s’agit d’une modification de la préposition ou conjonction par l’expression entre virgules et c’est la règle *link\_comma\_mod\_comma* qui établit les dépendances correspondantes.
* Au début d’une phrase, lorsqu’une conjonction de coordination est immédiatement suivie par une vrigule juste avant une expression dont la tête est un verbe fini, la règle *cc\_comma\_fin-verb* rattache le verbe fini à la conjonction par une dépendance ***dep.coord***.
* Lorsqu’au début d’une proposition, un arbre de dépendance est suivi par une pause, puis un autre arbre qui clôt une phrase et qui n’a pas de verbe fini comme racine, la règle *clstart\_cat\_pause\_cat\_stop* rattache la pause au second arbre de façon à ce que celui-ci vienne étendre sur la droite le premier.

**Constructions syntaxiques particulières**

***Citations***

Le module *quotes* traite des citations au sens large, c’est-à-dire des expressions entre guillemets, entre parenthèses ou entre crochets. Il est constitué d’une règle unique *parentheses* qui rattache un couple de guillemets, parenthèses ou crochets encadrant un arbre à cet arbre.

***Comparaisons***

On considère qu’une comparaison est une construction syntaxique formé de deux termes représentés par deux arbres T1 et T2.

Le module *comparative* traite de deux situations :

* Un arbre T1 est immédiatement suivi d’une conjonction de subordination « que », notée QUE, immédiatement suivie d’un arbre T2 dont la racine n’est pas un verbe fini.
* Un arbre T1 est immédiatement suivi d’un arbre de racine une conjonction de subordination « que » gouvernant un sous-arbre T2.

On va chercher le long de la frontière droite de T1 la racine TERM1 d’un sous-arbre correspondant au premier terme d’une comparaison. TERM1 est reconnu parce qu’il gouverne un adverbe ou un adjectif exprimant un degré dans une comparaison, ou qu’il est lui-même l’expression d’un degré.

Lorsqu’un tel terme est trouvé, on considère que le second terme de la comparaison est représenté par l’arbre T2 de racine TERM2.

La comparaison est établie par une dépendance ***arg.comp*** de TERM1 vers QUE et une dépendance ***obj.cpl*** de QUE vers TERM2, lorsque cette dernière n’existe pas.

Les règles de mise en œuvre utilisent un trait *compar* affecté à la conjonction « que » qui prend la valeur *test* pour exprimer que la conjonction doit être testée, la valeur *found* pour exprimer qu’une comparaison a été trouvée, et *fail* pour exprimer l’échec dans la recherche d’une comparaison.

Elles utilisent aussi deux dépendances provisoires qui ont comme cibles QUE, ROOT.PART qui a comme source la racine de l’arbre dont la frontière droite a été parcourue et GOV.PROV qui a comme source le mot testé comme TERM1.

Le trait *compar* n’est pas effacé à la fin du processus afin d’éviter une boucle infinie. Un module *comparative\_clear* le fait juste avant le module *comparative*.

***Clivées***

Lorsqu’une clivée est complétement analysée comme une proposition relative introduite par « qui », c’est une extension à droite d’un arbre qui permet de la rattacher au verbe « être » dont elle dépend.

Le module *cleft* traite des clivées avec complémenteur « que »[[3]](#footnote-3) .

Nous distinguons deux cas selon que le pronom relatif « que » a déjà trouvé son gouverneur ou pas. Le second cas se divise encore en deux, selon que « que » dépend directement ou indirectement du verbe tête de la clivée.

***Quiconque***

On considère que « quiconque » est à la fois sujet de la relative et de la principale dont elle dépend.

**Itération de l’analyse incrémentale**

L’analyse est menée de gauche à droite à partir de ce qui est considéré comme des débuts de phrases ou de propositions. Pour une raison ou pour une autre, l’analyse peut être stoppée. Dans ce cas, l’idée est de la reprendre à partir de l’endroit où elle s’est arrêtée et d’itérer ce processus autant de fois que nécessaire. Le point de départ de chaque analyse est marqué par un token vide inséré dans l’énoncé à analyser. Les points de départ des différentes analyses sont reliés entre eux par des dépendances ***DEP.START***. Une dépendance ***DEP.START*** a comme source un point de départ et comme cible la point de départ de l’analyse suivante.

Le module *add\_parsing\_start* calcule le point de départ d’une nouvelle analyse. Il le fait en parcourant l’énoncé de gauche à droite à partir du point de départ de l’analyse précédente suivant les dépendances SUC. On s’arrête quand on a trouvé un token qui remplit les conditions suivantes :

* Il n’est pas un début de phrase.
* Il n’est pas la racine d’un arbre qui suit immédiatement un début de phrase.
* Il n’est pas la racine d’un arbre qui suit immédiatement le point de départ de l’analyse précédente.
* Il n’est pas un signe de ponctuation à l’exception d’un guillemet ouvrant.

Dès qu’on a trouvé un tel token, on introduit un token vide à sa gauche immédiate qu’on relie par une dépendance ***DEP.START*** au point de départ de l’analyse précédente.

Ce token constitue le point de départ d’une nouvelle analyse qui est reprise à l’aide des modules destinés à être itérés.

**Mises à jour générales en fin d’analyse**

La réécriture de graphes permet de prendre des décisions provisoires en cours d’analyse qui peuvent être corrigées en fin d’analyse lorsque l’on dispose d’un maximum d’informations.

***Suppression des points de départ des analyses***

Le module *parsing\_start\_clean* supprime les tokens vides points de départ d’analyses et les dépendances ***DEP.START*** qui les relient entre eux.

***Correction des dépendances syntaxiques liées aux propositions relatives***

On effectue à l’aide du module *relative\_fix* les corrections dans trois cas :

* Le verbe tête de la relative a un infinitif objet direct et le pronom relatif est un modificateur de la tête.
* Le verbe tête de la relative a un complément qui est une infinitive introduite par une préposition et le pronom relatif « que » est un complément direct de la tête.
* Le verbe tête de la relative a un complément qui est une infinitive introduite par une préposition et le pronom relatif est un modificateur de la tête.

***Correction des dépendances argumentales des verbes et adjectifs***

Dans le cours de l’analyse, les verbes n’ont pas encore nécessairement tous leurs arguments et les arguments potentiels sont rattachés avec des dépendances provisoires. En fin d’analyse, on a toute l’information pour déterminer globalement l’ensemble des arguments de chaque verbe.

En cas d’ambiguïté, on privilégie les cadres de valence avec affixe (module *aff\_verb\_compl\_update*), puis ceux avec plusieurs arguments (module *verb\_compl2\_update*) pour terminer par les cadres avec un seul argument (module *verb\_compl1\_update*). Chaque règle correspond à un cadre de valence particulier.

Le module *adj\_compl\_update* met lui à jour les cadres de valence des adjectifs, chaque cadre donnant lieu lui aussi à une règle particulière.

***Correction des dépendances pour les pronoms, les auxiliaires et la ponctuation***

Le module *mod\_pro\_update* corrige les dépendances syntaxiques qui ont pour cible un pronom clitique « en » ou « y » ou le pronom relatif « dont ». Ces pronoms ont été systématiquement rattachés au verbe dont ils dépendent mais en réalité, ils dépendent d’un argument de ce verbe.

Le module *aux\_update* rétablir les auxiliaires modaux « devoir » et « pouvoir » comme verbes pleins.

Le module *punct\_update* modifie ou crée des dépendances relatives à la ponctuation.

***Correction des dépendances syntaxiques erronées***

Le module *synt\_correction* corrige les erreurs syntaxiques régulières.

* S’il y a une dépendance ***mod*** ou ***DEP.PROV*** du verbe « avoir » vers un participe passé, la règle *avoir\_update* considère qu’ « avoir » est un auxiliaire de temps du verbe au participe passé.
* S’il y a une dépendance ***mod*** d’un verbe à l’indicatif ou au subjonctif vers la tête d’un syntagme nominal qui le précède et si le verbe n’a pas de sujet, la règle *subj\_update* assume à ce groupe nominal le rôle de sujet.
* Si un groupe nominal de type locatif ou temporel est sujet d’un verbe et s’il gouverne un autre groupe nominal à l’aide d’une dépendance ***DEP.PROV***, ce second groupe nominal étant placé avant le verbe, alors la règle *temp-loc-subj\_update* transforme le premier groupe nominal en modificateur et le second en sujet.
* S’il y a une dépendance ***mod*** ou ***mod.voc*** d’un verbe vers un mot et une dépendance ***DEP.PROV*** de ce mot vers un adverbe, un syntagme prépositionnel ou une proposition subordonnée, la règle *mod\_update1* transforme la cible de ***DEP.PROV*** en modificateur du verbe.
* S’il y a une dépendance ***DEP.PROV*** d’un adverbe, une préposition, une conjonction de subordination vers un verbe, la règle inverse la dépendance avec le type ***mod***.
* Si un groupe nominal est un modificateur d’un verbe fini et s’il a lui-même un modificateur qui est un participe, la règle *mod\_part\_update* transforme le groupe nominal en sujet du participe et le participe en modificateur du verbe.

Le module *coord\_correction* corrige lui les erreurs relatives aux coordinations et énumérations.

***Décomposition des expressions figées***

Le module *verbal\_locution\_update* traite des locutions verbales qui, dans l’analyse, sont considérées comme des mots composés. Il réintroduit les dépendances syntaxiques standards entre les verbes têtes des locutions et leur complément.

Le module *frozen\_expr\_expand* rétablit les dépendances syntaxiques standard dans les mots composés qui sont des locutions prépositionnelles formées d’une préposition suivie d’un nom et d’une autre préposition ou d’une conjonction.

**Mises à jour spécifiques au schéma d’annotation de Sequoia**

Le schéma d’annotation de Sequoia a fait des choix discutables dont nous nous sommes affranchis dans notre système de règles. Néanmoins, à la fin de l’analyse, pour effectuer notamment des comparaisons, il est utile de transformer l’annotation afin de la rendre conforme au schéma de Sequoia.

Un premier module *syntax\_adapt* transforme un certain nombre de constructions syntaxiques, à savoir : noms modificateurs de nome en chaîne, coordination multiple, adverbes au superlatif, conjonctions de coordination se comportant comme des adverbes, auxiliaires à un temps composé, adverbes de quantité appliqués à des noms, dates, compléments d’origine, comparaisons, clivées.

Le module *phon\_fold* rétablit ensuite les contractions phonologiques. Le module *sequoia\_compos\_update* recompose les mots composés qui ne sont pas traités comme tels dans Sequoia. Le module *foreign\_expr\_update* traite des expressions étrangères.

Le module *punct* rétablit les dépendances de ponctuation telles qu’elles sont annotées dans Sequoia et le module *final\_punct* ajoute une dépendance de la tête de l’énoncé vers le signe de ponctuation final.

**Mise à jour de certains traits et suppression des dépendances de succession**

Le module *feat\_update* supprime les traits intermédiaires qui ont été utilisés par l’analyse et rétablit certains initiaux qui avaient été effacés.

Le module *clean* supprime à la fois les dépendances ***SUC*** et la marque artificielle de fin de l’énoncé, lorsqu’elle existe.

Le module *intermed\_synt\_dep\_clean* supprime les dépendances intermédiaires utiles pour l’analyse syntaxique.

1. La *couverture* d’un arbre est l’ensemble des mots qui sont des nœuds de cet arbre ; la couverture d’un nœud est la couverture de l’arbre qui a pour racine ce nœud. [↑](#footnote-ref-1)
2. On appelle *frontière droite* de l’arbre l’ensemble de tous ses nœuds N dont la couverture contient la partie à droite de N de la couverture de l’arbre [↑](#footnote-ref-2)
3. Dans Sequoia, il est considéré à tort comme un pronom relatif et non comme une conjonction de subordination. L’exemple « C’est à Paris que je vais » montre que c’est erroné. [↑](#footnote-ref-3)