Améliorations apportées à JSrealB, un réalisateur de texte français et anglais

Francis Gauthier
Département d'Informatique et de Recherche Opérationnelle
Université de Montréal

August 22, 2016

Résumé: Ce document est un rapport de stage écrit par Francis Gauthier au sujet de son stage d'été au laboratoire RALI de l'Université de Montréal, sous la direction de Guy Lapalme. Francis a travaillé sur la reprise du projet JSrealB qui avait été auparavant commencé par deux autres étudiants du RALI. Le but du stage était de peaufiner le programme et de lui ajouter plusieurs fonctionnalités qui n'avaient pas été encore implémentées.

Contents

1 Introduction

Au fil de son développement, JSrealB a connu plusieurs étapes. En 2014, Nicolas Daoust a commencé le projet en créant JSreal[1]. Il développa un réalisateur de texte facilement adaptable au web en langage JavaScript. Cette version ne traitait que le français. L'année suivante, Paul Molins a repris le projet et en a fait une version bilingue, en français et en anglais[4]. À l'aide d'une utilisation systématique de tables de règles et de lexiques annexes au programme, JSreal devint JSrealB. Le réalisateur de texte devint alors bilingue et prêt à accueillir d'autres langues semblables au français et à l'anglais. Après 2015 et après les modifications de Paul, le projet est laissé sur la glace avec plusieurs options à rajouter au programme. Entre autres, plusieurs types de phrase telles que les phrases interrogatives et négatives n'étaient pas implémentées, dans les deux langues. L'élision avait besoin d'être revue et les verbes aux temps composés n'étaient pas pris en charge. Francis Gauthier a donc repris le projet à l'été 2016. Ce rapport fait état des modifications qu'il a apportées à JSrealB.

2 Qu'est-ce que JSrealB?

La présente section présente le prototype de JSrealB tel qu'il était jusqu'en mai 2015 et le fonctionnement général du programme. Dans les sections suivantes(de 3 à 7) sont décrits les changements apportés au prototype.

JSrealB est un réalisateur de texte réalisant des phrases en français et en anglais. La fonction première du programme est de prendre en entrée une structure de données composée d'objets JavaScript qui représente une structure d'arbre syntaxique. JSrealB donnera en sortie la phrase correspondante à l'arbre donné en entrée. Parmi les parties complexes de la réalisation, on retrouve les accords en genre et en nombre entre les différents constituants de la phrase, l'application des différentes options prescrites par l'utilisateur, ainsi que la mise en forme de la phrase.

2.1 Les objets JSrealB

L'Exemple 1 démontre des appels à JSrealB et leur réalisations. À noter que JSrealB utilise une représentation de la structure d'une phrase en terme de ses constituants syntaxiques.

Les appels en majuscule représentent des fonctions qui créent des objets JSrealB. Ceux-ci sont classés en deux catégories.

Éléments terminaux Les éléments terminaux forment les mots de la phrase. Ils font habituellement partie des syntagmes. Ils peuvent être appelés sans syntagme, mais cela risque d'engendrer peu ou pas d'accords avec les autres constituants de la phrase. Le Tableau 1 liste les différents types d'éléments terminaux.

Exemple 1 Quelques utilisations de JSrealB

```
S(NP(D('le'),
     N('chat')),
  VP(V('aimer'),
     NP(D('le'),
        N('jouet'))))
```

Le chat aime le jouet.

```
S(NP(D('le'),
     N('souris')),
  VP(V('partir').t("f"),
     PP(P("avant"),
        NP(D('le'),
           N('chat').n("p")))))
```

La souris partira avant les chats.

Tableau 1: Les éléments terminaux

Appel JSrealB	Classe de mot	Exemple
`V`	Verbe	*aimer*
`N`	Nom	*chat*
`D`	Déterminant	*le*
`A`	Adjectif	*beau*
`Pro`	Pronom	*je*
`P`	Préposition	*de*
`Adv`	Adverbe	*très*
`C`	Conjonction	*et*

Syntagmes Les syntagmes sont des groupes d'éléments de la phrase. Ceux-ci sont des éléments passifs qui ne peuvent qu'exister s'ils ont des enfants. Ils sont essentiels à la réalisation, car ils fournissent l'information nécessaire pour comprendre les interactions entre les différents constituants de la phrase. Si on pense à un arbre syntaxique, ces syntagmes représentent les noeuds internes de l'arbre. La Tableau 2 liste les syntagmes disponibles.

Tableau 2: Les syntagmes

Appel JSrealB	Syntagme	Exemple
`S`	Phrase	*Le chat aime le jouet.*
`NP`	Syntagme nominal	*le chat*
`VP`	Syntagme verbal	*aime le jouet*
`AP`	Syntagme adjectival	*très beau*
`AdvP`	Syntagme adverbial	*bien heureusement*
`PP`	Syntagme prépositionnel	*à la maison*
`CP`	Syntagme coordonné	*la pomme, la poire et la pêche*

2.2 Les transformations

Chaque syntagme et élément terminal est un objet JavaScript, ce qui lui permet d'avoir des propriétés propres. Ces propriétés sont présentées au Tableau 3. Certains éléments recevront des propriétés par défaut du programme lors de l'exécution. C'est-à-dire qu'un nom aura comme genre par défaut le masculin et le nombre singulier. Un verbe recevra comme temps par défaut l'infinitif. Outre la valeur par défaut, il est aisé pour l'utilisateur d'en spécifier d'autres. Les méthodes spécifiant les propriétés voulues sont appelées directement sur les éléments terminaux en question.

Tableau 3: Les transformations principales

Propriété	Appel JSrealB	Exemple d'appel	Réalisation
Genre	`g`	`N("absent").g("f")`	*absente*
Nombre	`n`	`N("jeu").n("p")`	*jeux*
Personne	`pe`	`Pro("je").pe(2)`	*tu*
Temps	`t`	`V("aimer").t("i")`	*aimait*

Pour réaliser les différentes déclinaisons, le programme se base sur deux fichiers JSON préalablement générés automatiquement à partir du lexique exhaustif *dmf-lex* de la langue désirée. Il y a donc deux fichiers français, ainsi que deux fichiers anglais.

- Le premier est un **lexique** contenant les mots disponibles et le sigle de la table de déclinaison correspondante au mot.
- Le deuxième regroupe les **tables de règles** décrivant les différentes déclinaisons en genre et en nombre de chaque sigle, ou les conjugaisons en fonction des temps et des personnes pour les verbes.

Les syntagmes aussi peuvent subir des transformations. Par exemple, si on désire modifier le genre de tous les éléments d'un syntagme, il est possible de le spécifier sur le

syntagme en tant que tel et le programme propagera ces propriétés sur ses constituants enfants durant l'exécution. Nous aborderons justement la propagation dans la section suivante.

2.3 La propagation

Une des parties les plus importantes du programme est la propagation, qui permet de réaliser les règles d'accord entre les différents constituants. La propagation est le fait que certains constituants de la phrase auront des propriétés dictées par l'utilisateur ou par défaut et que ces propriétés doivent être partagées avec d'autres constituants de la phrase pour faire en sorte de bien accorder les différentes sections de la phrase. On note trois types de propagations différentes:

Propagation parent→enfant Lorsqu'une propriété est attribuée à un syntagme, on croit que l'utilisateur désire que cette propriété soit partagée entre tous ses constituants. Donc, pour chacun des enfants d'un syntagme parent, ils recevront les propriétés de leur parent immédiat.

Propagation frère/soeur À l'intérieur même d'un syntagme, on retrouve des éléments qui forment le noyau ou des éléments subordonnés au noyau. Dans ce cas, les propriétés du noyau seront propagées vers les éléments subordonnés pour que les éléments formés dans le syntagme soient tous accordés de la même façon. C'est cette propagation qui passera les informations du syntagme nominal sujet au syntagme verbal principal pour l'accord du verbe, par exemple.

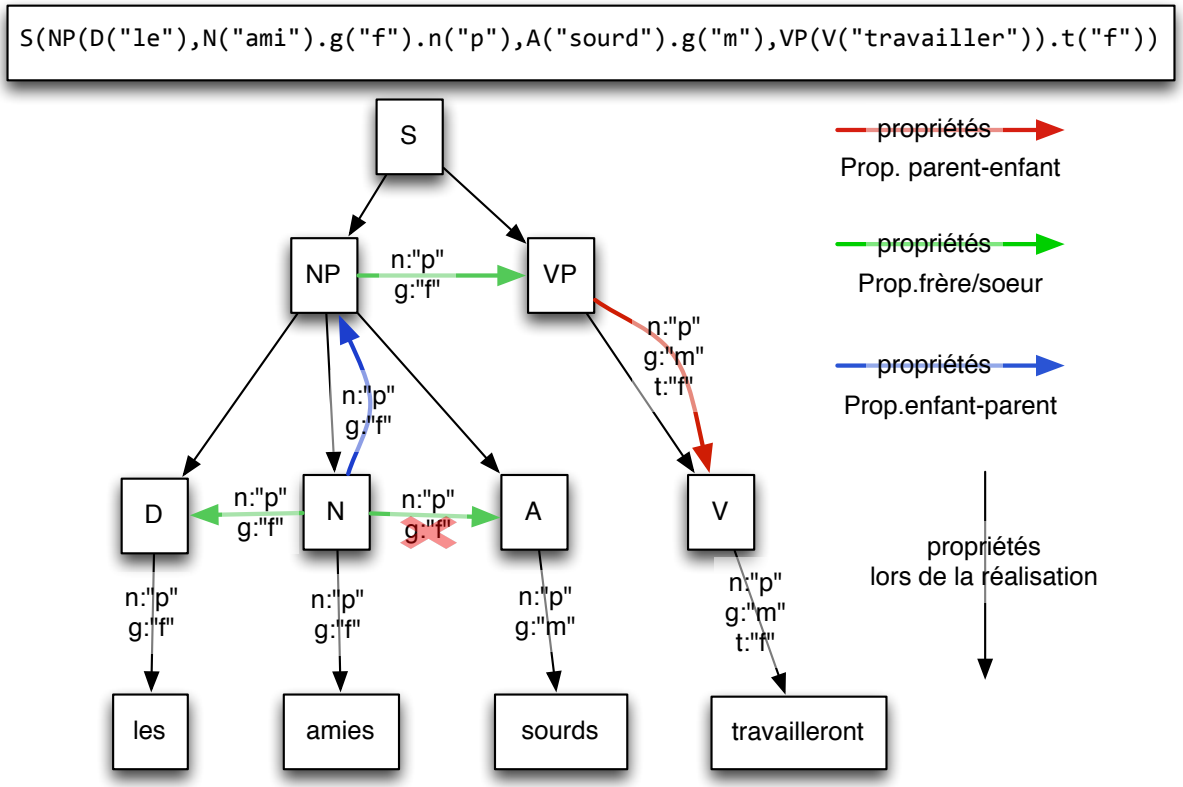
Propagation enfant→parent Certains noyaux nécessitent de transmettre leurs informations à leur parent pour que ceux-ci transmettent ensuite l'information ailleurs. C'est le cas pour le noyau du syntagme nominal(le nom) qui transmettra ses propriétés à son parent.

À noter que la propagation arrive en second plan face aux propriétés dictées par l'utilisateur. C'est-à-dire que les éléments possèdent en général des propriétés par défaut, qui peuvent être supplantées par les propriétés imposées par l'utilisateur. Ensuite, la propagation peut changer les valeurs par défaut, mais ne pourra pas changer une valeur imposée. La Figure 1 illustre la propagation sur les éléments et leurs réalisations.

L'encadré du haut représente l'appel JSrealB. Les éléments du bas sont ceux réalisés. La phrase finale serait: **Les amies sourds travailleront.* On voit dans le schéma les trois types de propagations, représenté par les flèches de couleurs. Il est intéressant de noter que la propagation du genre du nom vers l'adjectif n'aura pas lieu, car dans l'appel JSrealB, l'utilisateur a spécifié que le genre de l'adjectif devait être masculin. Cette spécification de l'utilisateur a priorité sur la propagation.

À noter que dans la majorité des cas, l'utilisateur ne spécifiera pas que quelques propriétés et que celles-ci seront propagées automatiquement et de manière à respecter les

Figure 1: La propagation



règles de la langue choisie. C'est-à-dire qu'un adjectif sera toujours bien accordé avec le nom auquel il est rattaché, à moins que l'utilisateur ne le spécifie autrement.

2.4 Autres fonctionnalités

JSrealB a quelques autres fonctionnalités. Il s'occupe notamment de la mise en forme des phrases. Il est possible de contrôler la mise en majuscule au début des mots, de rajouter de la ponctuation additionnelle et le programme comprend deux modules complémentaires très utiles, soit *Date* et *Number*. Ces deux modules permettent l'intégration des dates plus facilement à l'intérieur même des phrases, ainsi que pour les nombres.

Aussi, JSrealB permet l'intégration des balises HTML, donc son intégration est aisée sur le web. Le Tableau 4 démontre un exemple d'utilisation de JSrealB pour la cération d'une balise HTML.

Tableau 4: Utilisation d'un tag HTML

`N('joueur').tag('u')`	Appel
`<u>joueur</u>`	Réalisation JSrealB
joueur	Sur une page HTML

2.5 Exécution

Il est intéressant de comprendre le fonctionnement de JSrealB et les étapes principales depuis l'appel jusqu'à la réalisation finale.

Comme la structure JSrealB est arborescente, les étapes sont appelées récursivement sur les enfants de chaque noeud. La réalisation de l'arbre se fait par un passage en profondeur (enfants d'abord). Les étapes sont:

1. Classer ses différents constituants (modificateur, noyau, subordonnée ou complément)
2. Créer une liste ordonnée de ces constituants à réaliser
3. Réaliser chacun de ces constituants dans cet ordre
4. Concaténer chaque réalisation pour former sa propre réalisation
5. Appliquer des règles d'élision et de ponctuation sur sa réalisation

2.6 SimpleNLG-EnFr

Il existe un réalisateur de texte anglais et français précurseur à JSrealB. SimpleNLG-EnFr est une adaptation de SimpleNLG faite par Pierre-Luc Vaudry au cours de ses études au RALI

du département d'informatique de l'Université de Montréal[5]. Le réalisateur SimpleNLG est capable de générer un éventail plus grand que JSrealB. Son implémentation a été faite en Java. À la fin du projet de JSrealB, on s'attend à ce que les deux réalisateurs soient de même calibre. À cette fin, plusieurs des fonctionnalités de SimpleNLG-EnFr qui n'existaient pas dans JSrealB ont été travaillées. Le travail de Pierre-Luc Vaudry a servi de modèle pour orienter les modifications apportées à JSrealB.

3 Modifications apportées aux syntagmes

Après le développement de JSrealB jusqu'en 2015, la structure du programme était bien implantée, mais plusieurs fonctionnalités étaient manquantes. La section suivante présente quelques modifications qui ont été faites au niveau des syntagmes.

Lors d'ajout aux possibilités de JSrealB, il était évident que certaines fonctionnalités allaient demander des ajouts mineurs, alors que plusieurs autres nouvelles fonctions du réalisateur de texte nécessiteraient de déconstruire certaines sections de l'arbre syntaxique et possiblement de rajouter des éléments à cet arbre. Quatre options principales ont demandé à réarranger la structure de l'arbre syntaxique en cours d'exécution: la **phrase passive**, la **pronominalisation** d'un syntagme nominal, la **phrase impérative** et la **phrase interrogative**.

Par exemple, pour la phrase impérative, il faut s'assurer de retirer le sujet, si l'utilisateur ne l'a pas fait lui-même. Les différentes étapes citées à la Sous-section 2.5 seront interrompues dans le fil d'exécution après l'étape 3 pour réajuster la structure de l'arbre. Ensuite, on retournera à la racine de l'arbre pour recommencer la réalisation de départ. Cette étape est nécessaire, car, dans la phrase passive, le sujet et le complément d'objet direct sont inversés et il est utile de laisser JSrealB refaire la propagation du nouveau sujet du nouvel arbre pour éviter les conflits qui pourraient résulter d'une modification manuelle. Il est donc important de comprendre que pour une phrase avec peu d'options, soit une phrase assez simple, l'arbre sera traversé en profondeur pour accomplir la réalisation, mais que si les options précédentes sont présentes, la réalisation d'un arbre peut nécessiter plusieurs passages de l'arbre. Même une phrase assez complexe ne prendra que peu de passages, car les options sont assez limitées et certaines options sont incompatibles avec d'autres, notamment la phrase impérative, interrogative ou passive, qui sont incompatibles entre elles.

3.1 Syntagme nominal

3.1.1 Pronominalisation

La principale modification apportée à JSrealB dans le syntagme nominal est la possibilité de le pronominaliser. Par la pronominalisation, on veut faire l'action de prendre un groupe nominal et le transformer en un pronom. Par exemple, *le garçon* → *il*. La pronominalisa-

tion se fait essentiellement par la création d'un nouvel objet de type *pronom* qui héritera des propriétés qu'avait anciennement le syntagme nominal. Le positionnement de ce nouvel objet dans l'arbre syntaxique dépend de sa fonction dans la phrase. Si le syntagme original avait une fonction de

- **sujet** dans la phrase, il sera remplacé par un pronom personnel; *je* ou *I*, décliné à la bonne personne.
- **complément d'objet indirect**, on le remplace par un pronom personnel; *moi* ou *me,* décliné à la bonne personne.
- **complément d'objet direct**, on remplace le syntagme par le pronom *me*, en anglais. En français, ce sera le pronom *le*, tous deux déclinés selon la personne voulue.

Dans les deux premiers cas et dans le cas du complément d'objet direct **anglais**, les pronoms prennent la place du nom qu'ils remplacent.

Par contre, pour le complément d'objet direct **français**, c'est un peu plus complexe. On doit changer pour le pronom *le* décliné selon la personne, mais on doit placer le pronom **avant** le verbe. JSrealB devra donc modifier les éléments du syntagme verbal associé pour y changer la position du complément d'objet direct. Ce faisant, le complément d'objet direct se déplace avant le verbe et peut causer l'accord d'un participe passé employé avec *avoir*. La modification à la structure demandera de recommencer la réalisation de l'arbre initial pour le nouveau.

La pronominalisation du sujet fonctionne très bien, par contre on observe des problèmes pour l'objet indirect. Les phrases obtenues sont syntaxiquement correctes, mais diffèrent un peu du résultat attendu. Par exemple, en pronominalisant l'objet indirect de cette phrase: *allons à la maison* ou *go to the house,* on obtiendra *allons à elle* et *go to him*.

On s'attendrait à un pronom neutre en anglais comme *it*(*go to* ***it***), mais le genre par défaut est toujours masculin. Le changement vers un genre par défaut neutre offrirait certains avantages, notamment une meilleure pronominalisation, mais viendrait encombrer l'utilisateur qui voudrait appeler un pronom simple comme sujet. L'appel de `Pro("I")`, renvoie actuellement *he*, alors qu'avec le genre neutre par défaut, on obtiendrait *it*, ce qui n'est pas désiré, étant donné que le sujet est plus souvent une personne qu'un objet.

En français, on pourrait préférer *allons-y ou allons là-bas* à la place de *allons à elle*. L'utilisation de *là-bas* demanderait une connaissance sémantique, soit de savoir qu'on parle d'un lieu et l'utilisation du pronom *y* est restreinte à certaines prépositions du complément d'objet indirect. Ces cas sont rares et n'amélioraient que peu le programme par rapport à la difficulté engendrée pour repérer ces cas.

Pour ce qui est de la pronominalisation du **complément d'objet direct**, cela fonctionne assez bien, sauf lorsque d'autres options viennent modifier la structure de la phrase. Par exemple, la pronominalisation du complément d'objet direct avec un verbe négatif n'est pas encore pris en compte,comme le démontre Exemple 2.

Exemple 2 La pronominalisation et le négatif, simultané

```
S(Pro("je").pe(1),
  VP(V("voir").vOpt({neg:true}),
     NP(D("mon").pe(2),
        N("lunette").n("p"))))
```

Je ne vois pas tes lunettes.

```
S(Pro("je").pe(1),
  VP(V("voir").vOpt({neg:true}),
     NP(D("mon").pe(2),
        N("lunette").n("p")).pro()))
```

**Je les ne vois pas.*

3.2 Syntagme verbal

3.2.1 Temps composés

Au moment de sa reprise, JSrealB était en mesure de conjuguer les verbes à leurs formes simples.Nous avons ajouté la conjugaison des temps composés. Cela impliquait d'ajouter un ou plusieurs auxiliaires et un participe qui devra aussi s'accorder en français.

Commun aux deux langues Dans les deux langues, on retrouve des auxiliaires, principalement *avoir* et *être*. Les deux langues ont des participes: présent et passé. Chaque verbe présent dans le lexique est lié à une table de règles qui permet de conjuguer les temps simples facilement. D'ailleurs, l'accord du participe passé ou présent est indiqué dans les tables. Ces fonctionnalités seront exploitées pour réaliser les temps composés, mais de manière assez différente entre les deux langues.

Français Le français contient huit temps simples, formés d'un seul verbe accordé fourni dans les tables de règles. Ceux-ci sont:

- Présent de l'indicatif
- Imparfait
- Impératif présent
- Futur simple
- Passé simple
- Conditionnel présent
- Subjonctif présent
- Subjonctif imparfait

Seuls les temps subjonctifs requièrent le préfixe *que*, mais autrement, ils se ressemblent tous dans leur manière de s'accorder. Il n'y a que deux auxiliaires possibles en français, soit *avoir* et *être*. Ces auxiliaires dépendent du verbe à conjuguer.

La première étape de l'implémentation a été l'ajout des auxiliaires au lexique qui ne comportait pas d'information sur les auxiliaires. Ces informations étaient disponibles dans les fichiers à partir desquels est généré le lexique. Il a donc fallu quelques lignes de code pour extraire cette information supplémentaire de nos ressources. À partir du moment où le lexique a été actualisé avec ces modifications, on retrouvait alors un attribut à chaque verbe dans le lexique. L'attribut *aux* du lexique a comme valeur possible `êt`:*être*, `av`:*avoir* ou `aê`:*être ou avoir*.

Les verbes qui comportaient deux auxiliaires, par exemple le verbe *changer,* sont accordés avec l'auxiliaire *avoir*. Moins de 3% des verbes présents dans le lexique ont cette propriété. Il est toutefois possible de forcer l'utilisation d'un auxiliaire autre que celui par défaut avec l'option `.aux("être")` ou `.aux("avoir")` sur un verbe ou un syntagme verbal.

En bref, pour accorder un temps composé, on appellera à nouveau la fonction de conjugaison, avec comme paramètres:

- l'auxiliaire tiré du lexique
- le temps de l'auxiliaire (tiré d'une table de règle)
- les éléments de propagation du verbe en question (personne, genre et nombre par exemple).

L'auxiliaire obtenu sera ensuite concaténé avec le participe passé du verbe. Le participe passé en français requiert parfois un accord, donc la fonction qui sera appelée ne sera pas celle de conjugaison générale, mais une fonction particulière pour l'accord du participe passé, qui sera décrite à la Sous-section 3.2.2. Plusieurs autres mots auxiliaires et adverbes peuvent se glisser entre ou autour du verbe et de son auxiliaire. Ceux-ci sont décrits à la section 4.

Appel Pour appeler un verbe à un temps composé, il suffit de placer le paramètre correspondant dans la propriété *temps* d'un verbe ou d'un syntagme verbal. Le Tableau 5 démontre l'utilisation des temps composés francophones.

Anglais Les temps simples, c'est-à-dire ceux qui ne requièrent pas d'auxiliaire sont le *present tense* et le *past tense*. Dans le module de conjugaison, on intégrera le *future tense* comme un temps simple, même si ce temps requiert l'utilisation de l'auxiliaire *will*. Il existe plusieurs verbes auxiliaires qui changent le sens du verbe en anglais. Les deux principaux auxiliaires sont *be* et *have*, mais on retrouve aussi *will* et *do* très souvent.

Une phrase complexe en anglais requiert l'utilisation de plusieurs verbes auxiliaires, comme dans l'exemple *I will not have been going*. Dans l'exemple précédent, *will*, *have*

Tableau 5: Les temps composés en français

Appel	Temps	Réalisation
`V("aimer").t("pc")`	Passé composé	*a aimé*
`V("aimer").t("pq")`	Plus-que-parfait	*avais aimé*
`V("aimer").t("fa")`	Futur antérieur	*aura aimé*
`V("aimer").t("cp")`	Conditionnel passé	*aurait aimé*
`V("aimer").t("spa")`	Subjonctif passé	*ait aimé*
`V("aimer").t("spq")`	Subjonctif plus-que-parfait	*eût aimé*

et *been* sont tous des verbes auxiliaires. En anglais, les verbes auxiliaires dépendent **du temps de verbe** et non pas du verbe en tant que tel. Les temps composés sont les temps *perfect*, soit:

- present perfect
- past perfect
- future perfect

Ces temps requièrent tous d'appliquer l'auxiliaire *have*(accordé selon le genre et la personne) et le participe passé(invariable). Par contre, en ajoutant soit la forme passive, négative ou progressive(dite *continuous* en anglais), les auxiliaires changeront dépendamment de la combinaison des options imposées sur le type de la phrase. Le fonctionnement sera détaillé à la Sous-section 4.6.

Appel Pour obtenir un temps composé, il faut ajouter la propriété `.perf(true)` à un verbe ou à un syntagme verbal, comme le démontre le Tableau 6.

Tableau 6: Les temps composés en anglais

Appel	Temps	Réalisation
`V("love").perf(true)`	Present perfect	*has loved*
`V("love").t("ps").perf(true)`	Past perfect	*had loved*
`V("love").t("f").perf(true)`	Future perfect	*will have loved*

3.2.2 Accord du participe passé

En français, le participe passé doit s'accorder en fonction de son auxiliaire.[2]

Accordé seul Comme dans le groupe nominal *les fenêtres ouvertes.* Dans ce cas, il faut propager les propriétés de genre et de nombre aux verbes dans le syntagme nominal. Ainsi, lors de l'appel de conjugaison du verbe au participe passé, les propriétés du noyau nominal sont passées en paramètre et un appel simple d'accord au participe passé réalisera l'accord en genre et en nombre.

Accordé avec *être* On s'assure d'abord que le verbe est dans une forme composée, à l'instar du participe passé employé seul. Tout comme employé seul, le participe recevra les informations du sujet pour s'accorder. Ensuite, comme le lexique contient maintenant les informations sur les auxiliaires de chaque verbe, il suffit de décliner le participe passé selon une table de règle.

Accordé avec *avoir* Il est très difficile d'accorder le participe, car il doit s'accorder en genre et en nombre avec son complément d'objet direct, si celui-ci apparaît avant le verbe. Le premier type de phrase où un tel accord est nécessaire est un syntagme nominal à l'intérieur duquel on trouvera un syntagme propositionnel complément au nom, par exemple: *Les maisons que nous avons rencontrées.* La première étape pour implémenter cet accord a été de mettre en place le syntagme propositionnel, décrit à la Sous-section 3.4. Ensuite, les informations du noyau (*maison*) sont propagées vers les éléments du syntagme verbal en fonction du premier pronom rencontré. C'est-à-dire que dans l'exemple plus haut, le pronom introduisant la proposition est *que,* et dans ce cas, la proposition aura comme complément d'objet direct le syntagme nominal auquel il réfère *(les maisons*). Dans le cas où le syntagme propositionnel est *qui,* comme dans l'exemple *Les garçons qui jouent des tours,* alors le syntagme nominal *les garçons* agit en tant que sujet à la proposition. La propagation transmettra les informations de genre et de nombre du sujet pour accorder le verbe en conséquence. Dans le cas où le premier pronom est *que,* les informations du complément d'objet direct sont passées en paramètres au module de conjugaison qui accordera le participe passé avec auxiliaire avoir, s'il y en a un.

Le deuxième type de phrase où le participe passé devait être accordé avec avoir est une phrase où un pronom fait office de complément d'objet direct placé avant le verbe, comme dans la phrase *les pommes, je les ai cueillies,* ou bien *je l'ai appréciée, ta soeur.* En prenant comme exemple la phrase *les pommes, je les ai cueillies,* il nous aurait fallu la structure présentée dans l'Exemple 3

Dans cette structure, le pronom *les* est inclus manuellement. Le lecteur sait que *les* a comme antécédent *les pommes*, mais pour un programme informatique, il n'existe pas de lien entre les deux.

Pour ne pas avoir à spécifier les antécédents, il est plus pratique d'utiliser une pronominalisation des syntagmes nominaux, discutée à la Sous-section 3.1.1. Celle-ci est très utile pour faire apparaître des pronoms avec les propriétés de leurs antécédents, car ils ont été construits à partir de ceux-ci. Lors de la pronominalisation, si le syntagme nominal est

Exemple 3 Utilisation simple d'un pronom

```
S(NP(D("le"),
     N("pomme").n("p")),
  Pro("je").pe(1),
  VP(Pro("le").pe(3).n("p"),
     V("cueillir").t("pc")))
```

**Les pommes, je les ai cueilli.*

dans le syntagme verbal, on considère qu'il est alors complément d'objet direct. En le pronominalisant, le pronom vient se placer avant le verbe, ce qui crée une situation où le participe passé devra s'accorder. C'est alors que JSrealB propagera les propriétés du pronom au verbe comme étant celles de l'objet direct lui étant associé.

En combinant ces deux types de phrases, JSrealB peut accorder le participe passé avec l'auxiliaire avoir dans la plupart des cas. Par contre, avec des phrases plus complexes ou des figures de style, il probable que la propagation ne se fasse pas correctement, surtout si l'accord du participe passé requiert une interprétation humaine. Il est d'ailleurs à noter que JSrealB ne réalise en ce moment que des phrases individuelles, donc si l'antécédent d'un pronom se trouve dans une phrase différente, il ne serait possible de les accorder que par le passage d'une variable globale. Voir la Sous-section 5.2, pour plus de détails.

3.2.3 Les verbes impératifs

L'utilisation du verbe à l'impératif était disponible dans la dernière version de JSrealB, mais peu complétée. Tout d'abord, un verbe appelé sans option précisée par l'utilisateur recevra la troisième personne du singulier comme valeur par défaut. Suite à une légère modification, celle-ci a été changée pour la deuxième personne du singulier dans le cas d'un appel à l'impératif pour éviter de générer des erreurs fréquentes.

Par la suite, un retrait du groupe sujet a été ajouté au programme. L'ajout a été fait dans l'optique d'un appel bouclé modifiant automatiquement le temps de verbe sur une phrase qui contenait auparavant un sujet. Le retrait du sujet de la structure de l'arbre syntaxique requiert de recommencer la réalisation de l'arbre, car cela pourrait entrainer un accord du verbe différent.

3.3 Syntagme adjectival

3.3.1 Positionnement de l'adjectif en français

Anglais Le positionnement de l'adjectif est toujours avant le nom. Aucune modification à JSrealB n'a été apportée de ce côté-là.

Français Le positionnement de l'adjectif se fait majoritairement après le nom. Par contre, avec plusieurs adjectifs communs, le placement se fait avant.[3, pp. 432-436] Une première approche à ce problème a été de créer une fonction qui compterait les syllabes des adjectifs pour en déterminer la position, car des adjectifs monosyllabiques se placent habituellement avant le nom, comme *long, court, gros ou vrai.* Par contre, cette approche n'était pas prometteuse, car plusieurs adjectifs ayant plusieurs syllabes se placent avant, comme *premier*, et d'autres courts se placent après, notamment les adjectifs de couleur. Une solution, moins systématique, mais fonctionnant beaucoup mieux a été de dresser une liste des adjectifs à placer avant le nom. Ensuite, un attribut `'pre'` a été manuellement ajouté à ces mots dans le fichier *dmf-lex.txt* utilisé pour générer le lexique. Lesdits adjectifs ont donc eu un nouvel attribut dans le lexique français de JSrealB. Ensuite, en ajoutant l'option `pos` à placer sur un adjectif, l'utilisateur peut modifier la position de l'adjectif à l'intérieur même du syntagme nominal associé. Si aucune option n'a été dictée par l'utilisateur, le programme choisira comme option par défaut de mettre l'adjectif après, à moins que celui-ci soit annoté comme étant un adjectif antéposé dans le lexique.

3.4 Syntagme propositionnel

3.4.1 Utilisation

Le syntagme propositionnel est utilisé pour exprimer une proposition relative. Dans JSrealB, les syntagmes propositionnels sont appelés à l'intérieur même d'un syntagme nominal lorsqu'ils sont complément du nom ou à l'intérieur d'un syntagme verbal comme complément du verbe. Les Exemple 4 et Exemple 5 sont des exemples d'utilisation du nouveau syntagme et comment il s'inclut dans la structure syntaxique JSrealB.

3.4.2 Propagation

Dans une structure JSrealB, le syntagme propositionnel ressemble à une structure de phrase, c'est-à-dire que la propagation entre les éléments intérieurs du syntagme se fait essentiellement entre le groupe sujet et le groupe verbal. Par exemple, dans les syntagmes prépositionnels *dont tu m'avais parlé* et *that we met*, le verbe s'accorde avec le pronom qui le précède, tout comme dans une phrase. En fait, la propagation entre ses éléments est identique à celle d'une phrase. La propagation est plus intéressante au niveau du parent du syntagme propositionnel. Dans le cas où le syntagme en question est complément du nom, il sera alors enfant d'un syntagme nominal. Alors, plusieurs cas de figure peuvent subvenir.

1. (*français)*Le mot introduisant la proposition est ***que***, alors le noyau du syntagme nominal est le complément d'objet direct de la proposition qui devrait normalement contenir un verbe. JSrealB créera un nouvel objet avec les propriétés du noyau et les donnera au syntagme prépositionnel enfant comme étant les informations sur

Exemple 4 Utilisation du syntagme propositionnel en français

Français	

`S(NP(D("le"),`
`  N("maison").n("p"),`
`  SP(Pro("que"),`
`  Pro("je").pe(1).n("p"),`
`  VP(V("rencontrer").t("pc")))))`
	`S(NP(D("le"),`
	`  N("gens").n("p"),`
	`  SP(Pro("qui"),`
	`  VP(V("danser")))))`

Les maisons ***que nous avons rencontrées****.*	*Les gens* ***qui dansent****.*

`S(NP(D("le"),`
`  N("fleur"),`
`  SP(Pro("dont"),`
`  Pro("je").pe(2),`
`  VP(Pro("me").pe(1),`
`  V("parler").t('pq')))))`
	`S(Pro("je").pe(1),`
	`  VP(V("vouloir"),`
	`  SP(Pro("que"),`
	`  Pro('je').pe(2),`
	`  VP(V('partir').t('s')))))`

La fleur ***dont tu m'avais parlé****.*	*Je veux* ***que tu partes****.*

le complément d'objet direct. Ces informations seront utilisées pour accorder le participe passé avec auxiliaire avoir, si c'est le cas.

2. (*français/*anglais)Le mot introduisant la proposition est ***qui/who***. Dans ce cas, la proposition introduit une action qui est effectuée par le noyau du syntagme nominal. Dans ce cas, les propriétés du noyau sont transmises au verbe de la même manière qu'habituellement entre sujet et verbe, donc le syntagme propositionnel recevra les informations de genre et de nombre du noyau.

3. (*français/anglais*)Le mot introduisant la proposition est autre que les précédents cités (cas par défaut). Dans ce cas, il n'y a pas de propagation entre le syntagme nominal et le syntagme propositionnel enfant.

Il est aussi possible que le syntagme parent soit un syntagme verbal, auquel cas il ne semble pas y avoir de propagation à faire, comme dans le cas no.3 .

3.5 Syntagme adverbial, prépositionnel et coordonné

Les syntagmes adverbial, prépositionnel et coordonné avaient bien été implémentés jusqu'à maintenant. Au cours de mon stage, je n'ai pas eu à effectuer de changement sur ces syntagmes.

Exemple 5 Utilisation du syntagme propositionnel en anglais

Anglais	

`S(NP(D("the"),`
`  N("house").n("p"),`
`  SP(Adv("that"),`
`  Pro("I").pe(1).n("p"),`
`  VP(V("meet").t("ps")))))`
 	
*The house **that we met**.*
	`S(NP(D("the"),`
	`  N("people").n("p"),`
	`  SP(Pro("who"),`
	`  VP(V("dance").t("p")))))`
	
	*The people **who dance**.*

`S(NP(D("the"),`
`  N("car"),`
`  SP(D("which"),`
`  Pro("I").pe(2),`
`  VP(V("buy").t("ps")))))`
 	
*The car **which you bought**.*	

4 Les différents types de phrases

JSrealB est maintenant doté d'options permettant de donner un mode spécifique à la phrase. Sans option, les phrases sont déclaratives, actives et positives, mais il est maintenant possible de réaliser des phrases exclamatives et plusieurs types de phrases interrogatives différentes. Les phrases peuvent être mises sous forme passive, progressive et même négative.

Exemple 6 Utilisation des types de phrases

```
S(NP(D("le"),
     N("chat")),
  VP(V("manger"),
     NP(D("le"),
        N("souris")))).typ(neg:true,pas:true,prog:false,int:true,exc:false)
```

Est-ce que la souris n'est pas mangée par le chat?

Pour spécifier les types de phrases voulus, il faut passer en paramètre un objet qui spécifiera la valeur des paramètres voulus. Le Tableau 7 démontre les différents types de phrases possibel et leurs appels. L'Exemple 6 montre un exemple d'utilisation des types.

À noter que les types sont tous facultatifs, donc les appels avec paramètre `false` ne sont pas nécessaires.

Tableau 7: Les différents types de phrases

Type	Valeur par défaut	Appel
Exclamative	`false`	`S(...).typ({exc:true})`
Interrogative	`false`	`S(...).typ({int:true})`
Négative	`false`	`S(...).typ({neg:true})`
Passive	`false`	`S(...).typ({pas:true})`
Progressive	`false`	`S(...).typ({prog:true})`

4.1 La forme exclamative

Commun aux deux langues L'option de la phrase exclamative est une option simple qui s'applique seulement sur un objet `S`, une phrase. La phrase exclamative ne change en rien la structure initiale de la phrase et n'ajoute que la ponctuation exclamative, soit le point d'exclamation.

Français Il est intéressant de noter qu'en français, il est possible d'avoir une phrase à teneur exclamative et interrogative à la fois, par exemple: *Qu'est-ce que tu fais?!* est une phrase acceptée en français. Pour l'exemple précédent, on devra appeler la phrase avec l'option suivante: `.typ({exc:true,int:true})` . À noter que la double ponctuation a dû être ajoutée manuellement au lexique pour être acceptée comme ponctuation acceptable.

Anglais En anglais, un tel usage ne semble pas être acceptable selon *Antidote*, donc si les deux options(exclamative et interrogative) sont demandées, le type exclamatif sera ignoré et la phrase se terminera par un point d'interrogation.

4.2 La forme interrogative

La phrase interrogative est beaucoup plus complexe. Tout d'abord, dans la fonction de mise en forme de la phrase, la ponctuation ajoutée sera le point d'interrogation. Par contre, avant d'en arriver là, la réalisation de la phrase devra passer par une série de tests pour savoir si la structure de la phrase doit être changée. Cela est dû au fait que plusieurs types de questions peuvent être spécifiées. Les différents types disponibles sont démontrés dans le Tableau 8.
Le passage à la forme interrogative se fait en deux temps. D'abord, si la structure de la phrase a à être modifiée, elle le sera, et ensuite on ajoutera un préfixe à la phrase en tant que telle, par exemple *est-ce que* ou *where* par exemple.

Tableau 8: Utilisation de la phrase interrogative

Appel de l'option	Effet -*Exemple*
`.typ({int:'yon'})` `.typ({int:true})`	Question de type général. **Est-ce que** telle action est arrivée? *Est-ce que le chat mange la souris?*
`.typ({int:'wos'})`	Question par rapport au sujet(qui). **Qui** a fait l'action? *Qui mange la souris?*
`.typ({int:'wod'})`	Question par rapport à l'objet direct. **Qui** a subi l'action? *Qui est-ce que le chat mange?*
`.typ({int:'woi'})`	Question par rapport à l'objet indirect. À **qui** a-t-on fait l'action? *À qui est-ce que le chat mange la souris?*
`.typ({int:'wad'})`	Question par rapport à l'objet direct. **Quoi** a subi l'action? *Qu'est-ce que le chat mange?*
`.typ({int:'whe'})`	Question par rapport à l'endroit. **Où** s'est déroulé l'action? *Où est-ce que le chat mange la souris?*
`.typ({int:'how'})`	Question sur la manière. **Comment** s'est déroulée l'action? *Comment est-ce que le chat mange la souris?*

Commun aux deux langues Dans tous les cas, le programme fera un test sur la valeur du paramètre `int`. Dans plusieurs cas, la structure de la phrase sera modifiée. Bien évidemment, on peut passer l'option d'avoir une phrase interrogative à une phrase déjà bien formée. Par contre, cela implique que la réponse à la question est possiblement à l'intérieur même de la phrase. Il faudra donc effacer cette partie de la phrase pour que celle-ci ait du sens par la suite. Par exemple, si on désire avoir une question par rapport à l'objet direct(quoi), sur la phrase suivante: *Le chat a mangé la souris,* la question désirée est: *Qu'est-ce que le chat a mangé?* Donc, le programme JSrealB devra donc cerner la partie faisant office d'objet direct(*la souris*, dans l'exemple) et la retirer de la question.Pour cela, il est très important que la structure d'arbre ait été bien construite par l'utilisateur. C'est-à-dire qu'on trouvera le sujet d'une phrase comme étant le premier syntagme nominal ou le premier pronom de la phrase, on trouvera l'objet direct comme étant le syntagme nominal enfant du syntagme verbal principal et de manière similaire l'objet indirect comme étant le groupe prépositionnel issu du syntagme verbal principal. Si l'arborescence n'a pas été construite de la manière attendue, le programme ne réagira pas de la bonne façon et c'est valable pour beaucoup d'autres options. Les options diffèrent légèrement entre les deux langues.

Français Pour les différentes options, il y a quatre cas de figure différents. La troisième colonne représente respectivement les préfixes qui seront ajoutés par la suite selon le type d'option. Les préfixes sont tirés d'une table de règle, démontrée dans le Tableau 9

Tableau 9: Les différentes options de la phrase interrogative

Option interrogative	Retrait du constituant	Préfixe
`true/yon`	Aucun	*Est-ce que*
`wos`	Sujet	*Qui est-ce qui*
`wod`	Objet direct	*Qu'est-ce que*
`wad`	Objet direct	*Qui est-ce que*
`woi`	Objet indirect	*À qui est-ce que*
`whe`	Aucun	*Où*
`how`	Aucun	*Comment*

Anglais En anglais, le retrait des constituants ce fait de manière similaire, par contre c'est un peu plus complexe, car on doit parfois rajouter l'auxiliaire *do,* qui doit s'accorder en fonction du sujet, ce qui n'était pas le cas avec les préfixes français. Par exemple, *the dog runs* deviendra *does the dog run?,* donc on doit ajouter l'auxiliaire *do* et faire en sorte d'avoir le verbe à l'infinitif par la suite. Si le verbe appelé est au passé, alors l'auxiliaire *do* sera accordé au passé(*did*) et si le temps de verbe est le futur, l'auxiliaire sera *will* au lieu de *do.* En anglais, les différentes options sont listées dans le Tableau 10.

Tableau 10: Les différentes options de la phrase interrogative

Option interrogative	Retrait du constituant	Préfixe
`true/yon`	Aucun	*Do/will*(accordé)
`wos`	Sujet	*Who*
`wod`	Objet direct	*Who*+*do/will*(accordé)
`wad`	Objet direct	*What*+*do/will*(accordé)
`woi`	Objet indirect	*To who*+*do/will*(accordé)
`whe`	Aucun	*Where*+*do/will*(accordé)
`how`	Aucun	*How*+*do/will*(accordé)

4.3 La forme négative

Commun aux deux langues Essentiellement, l'ajout de la forme négative à la phrase nécessite l'ajout d'adverbes de négation autour du verbe dans les deux langues. Ceux-ci sont intégrés aux règles de chacune des langues. Par contre, leur position et interaction avec les autres types de phrases diffèrent fortement.

Français La négation se fait par l'ajout de deux adverbes en français. Le *ne* et le *pas*, principalement. Il est possible d'avoir un deuxième adverbe différent de *pas*, c'est pourquoi

il est possible de préciser le second adverbe de négation en français dans JSrealB, par contre, seule une liste restreinte d'adverbes sont acceptés, notamment *aucun, plus,rien, guère, etc...* En général, l'adverbe *ne* se place devant le verbe et l'adverbe *pas,*après. Par contre, lors de l'utilisation d'un temps composé, le deuxième adverbe de négation se place entre l'auxiliaire et le participe passé. C'est, entre autres, une des raisons pourquoi la négation est gérée dans le module de conjugaison et non pas ailleurs. L'Exemple 7 démontre quelques exemples de réalisations de phrases négatives.

Exemple 7 Utilisation de l'option négative

```
S(Pro("je"),
  VP(V("trouver").t("p"))).typ({neg:true})
```

*Il **ne** trouve **pas**.*

```
S(Pro("je"),
  VP(V("trouver").t("pc"))).typ({neg:true})
```

*Il **n'**a **pas** trouvé.*

```
S(Pro("je"),
  VP(V("trouver").t("pc"))).typ({neg:"rien"})
```

*Il **n'**a **rien** trouvé.*

Anglais En anglais, il n'y a qu'un adverbe de négation *not.* L'adverbe de négation est habituellement placé entre un premier verbe auxiliaire et le participe, comme dans *I will not love.* Pour les temps simples (present et past), il n'y a pas d'auxiliaire, mais on doit rajouter l'auxiliaire *do/did* dans ce cas. Par exemple, *I do not love you.* Finalement, le verbe *to be*, représente une exception, au sens où il ne nécessite pas l'auxiliaire *do* aux temps simples. Dans ce cas précis, l'adverbe de négation se place à la suite du verbe, comme dans *I am not.* Cette exception à la langue est une des rares exceptions que JSrealB prend en compte, car on s'attend à ce que son utilisation soit fréquente.

*À noter que la contraction de *do not* vers *don't* n'est pas prise en charge, étant donné que la première forme est toujours acceptée. Un ajout aux fonctions d'élision pourrait possiblement ajouter cette option.

4.4 La forme passive

Commun aux deux langues La phrase doit modifier sa structure pour accommoder le mode passif. Le programme réagira au fait que l'option passive a été déclarée en inversant le complément d'objet direct du verbe avec le sujet.

Il est possible que le sujet ou le complément d'objet direct ou les deux soient absents, auquel cas il n'y aura qu'un déplacement d'une section de la phrase au lieu d'une inversion.

Dans tous les cas, si la phrase avait un sujet, alors celui-ci devra être introduit par la préposition *par/by* lorsque le sujet deviendra l'objet indirect du verbe.

Étant donné que le sujet a possiblement changé suite à cette inversion, il est essentiel de reprendre la propagation pour que de nouveaux accords se fassent en fonction de la nouvelle phrase. JSrealB utilisera une fonction pour réinitialiser les propriétés héritées des noeuds parents ou frères/soeurs de l'arbre. Ensuite, la réalisation de l'arbre recommencera du début pour rétablir la bonne propagation entre les nouveaux constituants de la phrase.

Français D'abord, le verbe doit être accordé différemment. Principalement, les verbes avec l'auxiliaire *avoir* s'accorderont désormais avec l'auxiliaire *être.* Aussi, le verbe sera accordé au participe passé.

Exemple 8 Utilisation de l'option passive

```
S(NP(D("le"),
     N("chat")),
  VP(V("manger"),
     NP(D("le"),
        N("souris"))))
```

Le chat mange la souris.

```
S(NP(D("le"),
     N("chat")),
  VP(V("manger"),
     NP(D("le"),
        N("souris")))).typ({pas:true})
```

La souris est mangée par le chat.

Dans l'Exemple 8, dans la première phrase, *le chat* était sujet, alors que dans la deuxième phrase, il devient complément d'objet indirect. *La souris*, quant à elle, est passée de complément d'objet direct à sujet de la phrase.

Anglais Voir la Sous-section 4.6.

4.5 La forme progressive

Commun aux deux langues La forme progressive spécifie que l'action est en cours, en ce moment même.

Français Il suffit de rajouter le groupe de mots *en train de*. Dans le module de conjugaison, ce petit groupe de mots est inséré habituellement entre l'auxiliaire et le participe passé (s'il y en a un). Si le verbe n'est pas à un temps composé, il faudra rajouter un auxiliaire *être* pour obtenir la phrase *il est en train de manger*, par exemple. Il est aussi alors nécessaire d'accorder le verbe participe à l'infinitif. Toutes ces particularités sont gérées dans le module de conjugaison et peuvent dépendre fortement des autres options de verbe pour occasionner des cas légèrement différents.

Anglais L'ajout de la forme progressive ne requiert pas d'ajout d'un groupe de mots comme en français, mais changera l'accord du verbe. Voir la Sous-section 4.6.

4.6 L'accord du verbe en anglais, en fonction du type de phrase

L'utilisateur peut demander à ce que le verbe soit accordé au temps *perfect*, que la phrase soit passive et/ou progressive. Ces trois options s'influencent entre elles pour changer le résultat de la réalisation du verbe. Lors d'un appel à la réalisation d'un verbe une ou plusieurs de ces options, la réalisation se fera progressivement par des appels récursifs aux fonctions de conjugaison du module de conjugaison de JSrealB. En anglais, le fonctionnement est différent de celui en français qui passe plutôt par plusieurs tests, sans trop de récursivité. Pour chaque option de verbe activée, le programme ajoutera un auxiliaire et un participe dans les temps prescrits par la table des règles qui fût ajoutée aux ressources de JSrealB. Ceux-ci sont listés dans le Tableau 11:

Tableau 11: Les options de verbe anglais et leur participes

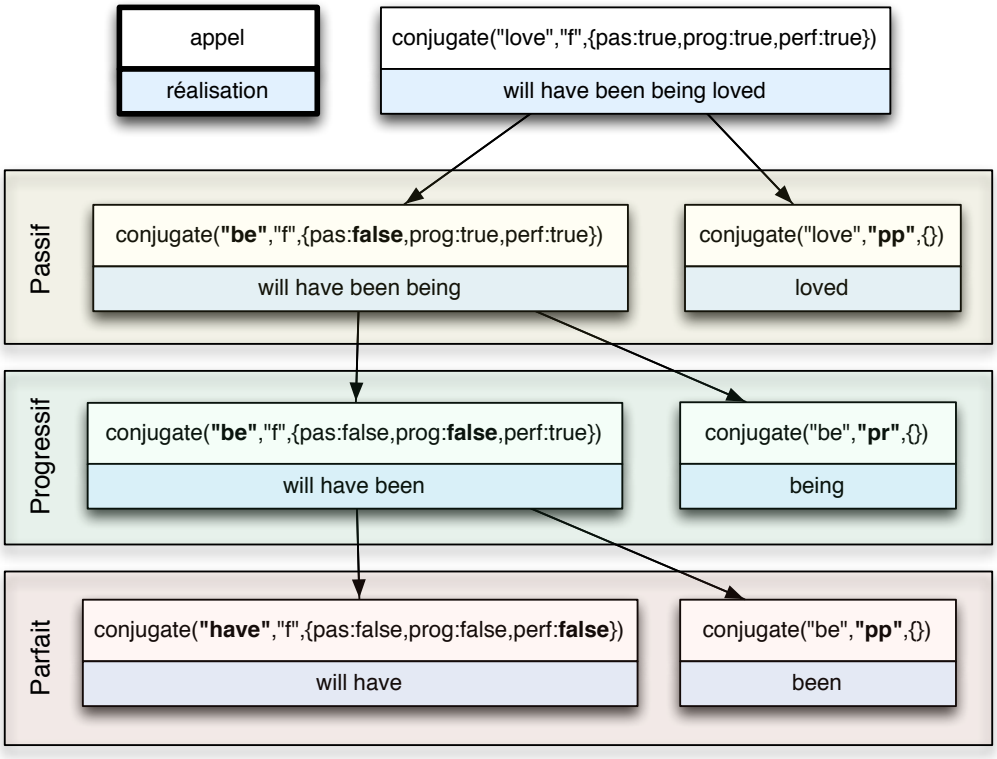
Option de verbe	Auxiliaire	Participe
passive (`pas`)	*be*	passé
continuous (`prog`)	*be*	présent
perfect (`perf`)	*have*	passé

Pour que la réalisation se fasse, le programme vérifie, dans l'ordre suivant: *passif,progressif* puis *perfect*, si une de ces options est activée. Si l'option est activée, le résultat de la réalisation de cette étape du verbe sera la concaténation du verbe auxiliaire au temps prescrit pas l'appel initial du verbe(present, past ou future) ainsi que de l'appel du verbe initial au participe passé/présent selon la règle. Ensuite, l'option qui vient d'être utilisée sera désactivée et les options de verbe résultantes seront passées en paramètres pour l'accord de l'auxiliaire. Celui-ci pourrait encore avoir des options de verbe à consommer, donc il devra refaire l'étape décrite récursivement, mais avec une option de verbe différente. Figure 2 explique la réalisation du verbe *to love*, avec les trois options de verbe initialement activées.

Figure 2

Exemple de conjugaison de verbe composé anglais et de ses options.

```
fonction conjugate(verbUnit, verbeTense, verbOptions)
```



Dans la Figure 2, l'appel initial se trouve au sommet de l'arbre. Les cases enfants représentent les appels récursifs qui sont appelés par le programme. Si on observe les options de verbe, elles sont les trois *true* initialement. Lors de chaque appel récursif, on observe que les appels de gauche ont une option consommée(qui est devenue *false* par rapport à leurs parents). Les trois options sont consommées dans l'ordre précisé plus tôt. Le verbe prescrit par l'option est appliqué à l'auxiliaire(gauche), alors que le participe obtient le verbe principal. Le temps de l'auxiliaire reste celui du verbe principal alors que le temps du participe est déterminé par l'option consommée. Au final, la récursivité se termine lorsqu'il n'y a pas d'option à consommer, donc en présence d'un temps simple (present, past ou future). Il reste qu'à remonter l'arbre et à concaténer les résultats pour obtenir le verbe désiré.

4.7 Restrictions

Si l'utilisateur le désire, les cinq formes de phrases différentes peuvent toutes être appelées simultanément. Par contre, certaines de ces combinaisons peuvent entrainer des résultats différents de ceux espérés, principalement parce que ce sont des cas qui n'existent pas dans la langue désirée. Voici quelques exemples d'appels/combinaisons qui peuvent ne fonctionneraient pas:

1. L'appel de la phrase passive sur un verbe français avec l'auxiliaire *être*
2. L'utilisation d'un verbe impératif avec:
   (a) Une phrase interrogative
   (b) Une phrase progressive

En réaction aux restrictions JSrealB réagira à ces cas sans lancer d'erreur fatale. Par contre, dans la plupart des cas, la phrase en sortie en sera probablement une qui n'est pas acceptée dans la langue utilisée. Sinon, JSrealB décidera d'ignorer une des options demandées pour avoir en sortie une phrase correcte.

5 Autres modifications

5.1 Liaison par le trait d'union

Parfois, la mise en forme de la phrase nécessite des traits d'union entre certains mots. C'est plus fréquent en français, majoritairement lorsque le sujet et le verbe sont inversés. Les inversions sont utiles pour plusieurs phrases interrogatives, mais l'approche utilisée et décrite à la Sous-section 4.2 permet de former des questions sans inversion du sujet et du verbe. Il y a aussi des cas où on pronominalisera le complément d'objet direct avec un verbe impératif, par exemple: *Mange-la!, Retrouve-moi là-bas.* Dans ces cas précis, JSrealB

ne fait pas encore la liaison et l'inversion, mais il est possible de le faire manuellement avec une option `.lier()`, disponible avec tous les mots. En appelant cette option, l'élément qui appellera la méthode `lier` sera connecté avec le prochain mot par un trait d'union. C'est un ajout simple à JSrealB qui comble un manque en français, mais qui permet aussi de créer des liaisons dans certains mots anglais aussi qui le nécessite, notamment: *well-known* ou *blue-green*. L'Exemple 9 présente l'utilisation de la liaison dans deux contextes.

Exemple 9 Utilisation de la liaison forcée

```
VP(V("rejoindre").t("ip").lier(),
   Pro("moi").pe(1),
   Adv("là-bas"))
```

rejoins-moi là-bas

```
NP(D("a"),
   A("blue").lier(),
   A("green"),
   N("algea"))
```

a blue-green algea

5.2 Clone et déréférencement

JSrealB a été spécialement développé pour fonctionner aisément avec le web notamment par le fait qu'il est écrit en JavaScript et qu'il y a des options pour intégrer des balises HTML aux réalisations directement. Il est donc normal de vouloir utiliser du code JavaScript pour simplifier la création d'objets JSrealB. Par exemple, dans la phrase *Les pommes, je les ai cueillies.* il faudrait appeler la phrase avec cet appel:

Exemple 10

```
S(NP(D("le"),
     N("pomme").n("p")).a(","),
  Pro("je").pe(1),
  VP(V("cueillir").t("pc"),
     NP(D("le"),
        N("pomme").n("p")).pro()))
```

Les pommes, je les ai cueillies.

Par contre, on appelle deux fois le même syntagme nominal. On pourrait utiliser une variable JavaScript pour éviter d'avoir à réécrire le syntagme 2 fois, ainsi: *var pom =* `NP(D("le"),N("pomme").n("p"))`. Par contre, l'appel de la phrase précédente en remplaçant par la nouvelle variable ne donnera pas le résultat voulu. L'Exemple 11 montre une utilisation simple d'une variable JavaScript.

Exemple 11

```
S(pom,
  Pro("je").pe(1),
  VP(V("cueillir").t("pc"),
     pom.pro()))
```

Elles, je les ai cueillies.

Malheureusement, dans l'Exemple 11, lors des deux appels de **pom**, les deux réalisations seront similaires, car les deux méthodes pointent vers le même objet. Lorsque le deuxième appel utilise la méthode `.pro()`, le syntagme obtient la propriété de pronominalisation, donc les deux appels de **pom** seront pronominalisés. Ce n'est pas le but désiré ici, donc JSrealB a maintenant une option pour créer un clone d'un objet JSrealB. Le clone hérite des propriétés initiales de l'objet, notamment son genre, son nombre, sa personne, etc. Par contre, à partir du moment où il est cloné, le nouvel objet ne partage plus ses propriétés avec un autre appel du syntagme original. Le Tableau 12 démontre l'utilisation de l'option `clone()` et l'Exemple 12 démontre l'utilité de cette fonction pour réaliser correctement notre exemple précédent.

Tableau 12

var `pom = NP(D("le"),N("pomme"))`	*la pomme*
`pom.clone.pro()`	*elle*
`pom.clone.n("p")`	*les pommes*
`pom`	*la pomme* (sans influence des appels précédents)

Exemple 12

```
S(pom,
  Pro("je").pe(1),
  VP(V("cueillir").t("pc"),
     pom.clone().pro()))
```

Les pommes, je les ai cueillies.

5.3 Élision

JSrealB a un module d'élision assez simple qui fonctionnait dans la plupart des cas de figure. En plus de cela, quelques modifications simples ont été ajoutées.

Français L'élision fonctionnait bien, sauf pour le pronom/déterminant *ce*. Celui-ci doit s'élider lorsqu'il est devant le pronom *en* ou devant une forme simple ou composée du verbe *être* commençant par une voyelle. Lorsque le module d'élision est appelé, la structure JSrealB n'est plus et les constituants de la phrase ont déjà été convertis en *string*. Cela fait en sorte qu'il n'est pas possible de vérifier directement la classe du mot, ni sa forme de base pour savoir si celle-ci est *en* ou *être*.

Pour contrer cela, une table contenant le pronom *en* et les formes du verbe *être* commençant par une voyelle a été ajouté aux ressources. Lorsque le mot à élider est *ce* et que le *token* le précédant est présent dans la table en question, *ce* sera élidé en *c'*.

Autrement, *ce* est élidé en *cet* devant un nom commençant par une voyelle. Finalement, dans les autres cas, il restera le même.

Anglais Devant un mot commençant par une voyelle, le déterminant *a* devient *an*. La règle stricte spécifie que l'élision doit se faire si le mot suivante a sa première syllabe qui sonne comme une voyelle. Donc, devant le mot *unique* par exemple, le *a* devrait rester tel quel.

Exemple 13

```
NP(D("a"),
   A("unique"),
   N("scene"))
```

**an unique scene* (JSrealB) *a unique scene*(correct)

Ce n'est pas le cas dans JSrealB. Le programme aurait besoin de connaître la phonétique des syllabes pour être capable de discriminer efficacement, ce qui n'est pas le cas. Tout de même, en majorité, lorsque le mot commence par une voyelle, il devrait causer l'élision.

6 Résultats

Après toutes les modifications apportées à JSrealB, on peut maintenant dire que celui-ci est maintenant un réalisateur de texte français et anglais prêt à l'emploi. Grâce aux maints

ajouts au programme, la grande majorité des phrases que l'on retrouve dans les textes de tous les jours peuvent être correctement réalisées par JSrealB de manière rapide. Pour ce qui en est d'un ajout d'une troisième langue au programme, par l'ajout du lexique et des tables de règles, on obtiendrait rapidement la plupart des fonctionnalités de JSrealB. Par contre, si la langue est quelque peu différente au français ou à l'anglais pour ce qui est des accords, il faudrait apporter directement des modifications au programme. À noter aussi que le module de conjugaison est assez différent entre les deux langues, donc il est à prévoir des modifications de ce côté pour l'intégration d'une nouvelle langue, ainsi que dans la modification vers la forme interrogative, qui est implémentée de manière similaire, mais séparément dans le code, à cause de plusieurs différences notables. La plupart des autres fonctionnalités sont adaptables facilement à une autre langue future et ne requerront que peu ou pas de changements.

7 Limitations et travail futur

Il reste tout de même plusieurs fonctionnalités qui n'ont pas été implémentées dans JSrealB, soit parce que leur utilisation est moindre, ou que l'implémentation semblait complexe. On y retrouve:

- Le passage automatique de *he/she* vers *it* en anglais lors de la pronominalisation d'un objet ou d'un animal.
- La dénombrement: *the milk,* au lieu de *a milk. De l'eau*, au lieu de *une eau.* Aussi lié à l'utilisation de *how much vs how many* en anglais.
- L'inversion sujet-verbe: *Le garçon a aimé le jeu.* → *A-t-il aimé le jeu?*
- L'utilisation de la pronominalisation ainsi que de la négation, en même temps: *Je n'ai pas aimé le jeu* → *Je ne l'ai pas aimé.* En ce moment, *n'ai pas aimé* est la réalisation directe du verbe. Par la pronominalisation, on ajoute le pronom avant, alors qu'on devrait l'insérer quelque part à l'intérieur même de cette réalisation.
- La contraction en anglais: *do not* → *don't , I am* → *I'm , etc.*
- L'élision de *a* vers *an* dans les cas où la première syllabe sonne comme une voyelle, mais ne commence pas par une voyelle. Aussi dans le cas où ce n'est pas une voyelle, mais que la syllabe sonne comme une voyelle.
- La phrase interrogative avec *combien*
- Les restrictions sur les types de phrases combinées citées à la Sous-section 4.7

8 Conclusion

JSrealB est désormais un réalisateur de texte qui réalise bien plus qu'il ne le faisait avant. L'apport a été:

- Une légère modification au niveau du lexique(pour les adjectifs et les auxiliaires),
- Quelques ajouts au niveau des règles de chaque langue(participe, adverbes interrogatifs et négatifs...)
- Plusieurs ajouts au niveau de JSrealB directement:
  - Un syntagme propositionnel a été ajouté
  - Le module de conjugaison a été complètement changé
  - La propagation a été modifiée auprès de quelques syntagmes
  - Etc.

Pour s'assurer que le tout fonctionne bien, des dizaines de tests unitaires ont été ajoutés pour assurer la stabilité du programme lors d'ajouts futurs et les quelques démonstrations qui existaient déjà de JSrealB ont été mises à jour pour démontrer les nouvelles fonctionnalités du programme.

References

[1] Nicolas Daoust and Guy Lapalme. *JSreal: A Text Realizer for Web Programming*, chapter 21, pages 363–378. Number Text, Speech and Language Technology, Vol 48. Springer, 2014. Université de Montréal.

[2] Maurice Grevisse. *Savoir accorder le participe passé.* Éditions J. Duculot, S.A., Gembloux(Belgique), 1975.

[3] Maurice Grevisse. *Le Bon Usage.* Éditions Duculot, Paris-Gembloux, 11e edition, 1980. Grammaire française avec des remarques sur la langue française d'aujourd'hui.

[4] Paul Molins. JSrealB : Approche systématique pour la réalisation multilingue de textes. 2015. http://rali.iro.umontreal.ca/rali/sites/default/files/publis/Rapport%20Synthese%20PFE.pdf.

[5] Pierre-Luc Vaudry and Guy Lapalme. Adapting SimpleNLG for bilingual English-French realisation. In *A. f. C. Linguistics, ed., 14th European Workshop on Natural Language Generation*, pages 183–187. Sofia, Bulgaria, 2013.