Rapport du Projet 3A

Hugo Barreiro M1 MPRI
Sous la direction de Samuel Mimram

Septembre 2024 - Mars 2025

Table des matières

1 Introduction **3**

2 État de l'art **3**

3 Problème traité et solution **5**
3.1 Problème 5
3.2 Première approche 6
3.3 Orientation 6
3.4 Générateur 7
3.4.1 Sous-chaînes des mots 7
3.4.2 Sous-ensembles des sous-chaînes des mots 7
3.4.3 Expansions des mots 8
3.5 Implications 9

4 Statistiques **9**
4.1 Temps d'exécution 9
4.2 Suites de nombres 9

5 Conclusion **10**
5.1 Résultats 10
5.2 Limitations 10
5.3 Perspectives 11
5.4 Remerciements 11

Références **12**

Annexes **13**

1 Introduction

Le mathématicien *Terence Tao*, le 25 septembre 2024, publie sur son *Blog* un projet collaboratif visant à cartographier les relations entre différentes théories équationnelles de Magmas. En d'autres termes, il s'agit de déterminer toutes les implications engendrées par un ensemble donné d'équations.

On peut examiner manuellement toutes les implications entre les équations, deux à deux. Cela reste envisageable pour un petit ensemble d'équations, mais devient rapidement impraticable lorsque l'ensemble s'agrandit. En effet, *Terence Tao* avait initialement analysé à la main un ensemble de 11 équations avant de s'attaquer à un ensemble beaucoup plus vaste, composé de 4694 équations.

C'est ici qu'intervient l'aspect collaboratif du projet. Ainsi, chacun était libre de proposer ses solutions, à condition qu'elles soient rédigées en *Lean*, permettant ainsi de vérifier automatiquement leur correction. De plus, face au très grand nombre d'implications à considérer (22 028 942), il était essentiel de concevoir une solution automatisable et informatisée.

C'est alors que l'on s'est tourné vers la réécriture de termes, notamment la *Complétion de Knuth-Bendix*. Cette dernière permet de déterminer les équations impliquées par une équation donnée, tout en fournissant les étapes de calcul nécessaires pour parvenir à ce résultat. On peut ainsi envisager de générer directement les preuves en *Lean* pour chaque implication.

2 État de l'art

Afin d'apprendre les bases de la réécriture, je me suis intéressé au livre *Term Rewriting and All That* de *Franz Baader* et *Tobias Nipkow*.

La réécriture est une branche de l'informatique théorique qui s'intéresse à la transformation d'objets syntaxiques en appliquant une succession de règles. Cela peut concerner des mots, des termes, des lambda-termes, des programmes, des preuves, etc. Ainsi, la réécriture est utilisée dans des domaines tels que la preuve automatique, l'optimisation de programmes ou encore la sémantique formelle des langages.

Ce qui distingue la réécriture de la logique équationnelle, c'est que les équations sont orientées de gauche à droite. Autrement dit, on ne peut remplacer que le côté gauche par le côté droit, mais pas l'inverse. Cela constitue un modèle de calcul Turing-complet, proche de la programmation fonctionnelle.

Un système de réécriture est constitué d'un ensemble de générateurs et de règles de réécriture. Les générateurs définissent la forme des termes. Une règle de réécriture s'écrit sous la forme $l \to r$, ce qui signifie que l'on peut remplacer chaque occurrence de l par r dans un terme. Par exemple, pour les mots, si l'on dispose d'un alphabet $\Sigma = \{a, b, c\}$ et d'une règle $aa \to b$, on peut réécrire le mot *bcaacb* en *bcbcb*.

Cependant, un système de réécriture peut conduire à des réécritures qui ne terminent pas. En effet, si l'on dispose de la règle $a \to ab$ et que l'on applique cette règle au mot a, on obtient $a \to ab \to abb \to abbb \to \cdots$, ce qui montre que la réécriture ne termine pas. Malheureusement, la propriété de terminaison pour un système fini de réécriture est souvent un problème indécidable.

Pour garantir la terminaison, on peut essayer de construire un ordre strict bien fondé $<$ tel que, si $t \to t'$, alors $t > t'$. Un tel ordre est appelé ordre de réduction. Parfois, un ordre bien fondé peut aussi suffire. Par exemple, un ordre bien fondé connu pour garantir la terminaison dans la plupart des cas est l'ordre lexicographique. Dans le cas des mots, ce dernier consiste à orienter une règle vers le mot le plus court s'il existe une différence de longueur, sinon vers le mot le plus petit au sens du dictionnaire.

De plus, un système de réécriture n'est pas forcément confluent a priori. En effet, si l'on dispose des règles $ab \to a$ et $ba \to b$, on peut obtenir $aba \to aa$, mais aussi $aba \to ab \to a$, avec aa et a comme formes normales. Cela montre qu'un système de réécriture n'est pas toujours confluent.

Pour vérifier si un système est confluent, il suffit de vérifier sa confluence locale dans le cas où il termine (selon le lemme de Newman). Dans ce cas, tester la confluence est décidable : il suffit d'examiner un nombre fini de configurations appelées paires critiques. Par exemple, pour un système composé uniquement des règles $ab \to \epsilon$ et $ba \to \epsilon$, il faut vérifier la confluence locale pour les mots aba et bab.

Ensuite, on peut rendre un système de réécriture confluent s'il termine, grâce à la *Complétion de Knuth-Bendix*, qui repose en partie sur l'étude des paires critiques. Ainsi, tout système qui termine peut devenir confluent. Un système qui termine et qui est confluent est appelé convergent, c'est-à-dire que la réécriture termine et produit toujours le même résultat. On note que pour obtenir un système convergent grâce à la *Complétion de Knuth-Bendix*, il faut utiliser au moins un ordre bien fondé afin d'assurer la terminaison du système complété.

L'objectif est donc d'obtenir un système de réécriture qui termine afin de le compléter pour qu'il devienne confluent, et donc convergent. Comme mentionné précédemment, on peut tenter de construire un ordre de réduction. Si cela échoue, on peut avoir recours aux transformations de Tietze.

Il existe quatre transformations de Tietze :

1. Ajouter une règle de réécriture qui peut être déduite des règles déjà présentes dans le système.
2. Supprimer une règle de réécriture si elle est dérivable à partir des autres règles.
3. Ajouter un générateur (par exemple, dans le cas des mots, un nouvel élément de l'alphabet). Par exemple, si l'on dispose de l'alphabet $\Sigma = \{a, b\}$ et de la règle $aa \to b$, on peut ajouter c à l'alphabet et introduire la règle $aa = c$.
4. Supprimer un générateur s'il est dérivable à partir des autres générateurs.

3 Problème traité et solution

L'archive du code est accessible sur *GitHub*. Le calcul des paires critiques et le fonctionnement de la *Complétion de Knuth-Bendix* sont expliqués plus en détail en annexes.

3.1 Problème

L'objectif initial était de s'intéresser à la réécriture des termes pour répondre au problème posé par *Terence Tao*. Ainsi, nous avions décidé d'abord d'étudier la réécriture des mots afin de poser des bases solides en observant le comportement de la réécriture sur des objets plus simples que les termes. Néanmoins, cette étude s'est révélée plus subtile et intéressante que prévu. Ainsi, nous nous sommes concentrés uniquement sur la réécriture des mots, que nous allons présenter dans la suite.

Nous avons commencé par mettre en œuvre la *Complétion de Knuth-Bendix* et la génération du diagramme de Hasse résumant les implications entre les différentes équations. Notre démarche est la suivante :

Pour chaque équation de notre ensemble, nous formons un système composé uniquement de cette équation, que nous complétons ensuite à l'aide de la *Complétion de Knuth-Bendix*. Le système complété est alors utilisé pour identifier les équations impliquées. Pour cela, chaque côté de l'équation est normalisé en appliquant les règles de réécriture. Si les deux côtés normalisés sont identiques, alors l'équation est impliquée par le système de réécriture, et donc par l'équation d'origine.

Informellement, la *Complétion de Knuth-Bendix* peut être vue comme une procédure visant à ajouter au système de réécriture toutes les équations déductibles du système de base afin de rendre ce dernier convergent. Ainsi, en normalisant les deux côtés d'une équation à l'aide d'un système complété, on cherche à vérifier si l'égalité est obtenue en tenant compte de toutes nos hypothèses. Par ailleurs, puisque le système de réécriture est convergent, la forme normale est unique ; si les deux côtés ne sont pas égaux, on sait alors que l'équation n'est pas impliquée par le système.

Cependant, un problème peut survenir : la *Complétion de Knuth-Bendix* ne termine que si le système de réécriture orienté termine. Or, comme nous l'avons constaté, ce n'est pas toujours le cas. Par conséquent, si la complétion ne termine pas, il sera nécessaire de trouver des moyens de garantir sa terminaison. Une fois tous les systèmes complétés, il ne restera plus qu'à générer le graphe des implications.

Nous nous concentrerons plus particulièrement sur l'ensemble des mots de longueur au plus 4, avec l'alphabet $\{a, b\}$. De plus, nous considérerons les équations à symétrie et à renommage près.

3.2 Première approche

La première étape a consisté à implémenter la *Complétion de Knuth-Bendix*, sans adopter de techniques particulières pour l'orientation ou les générateurs. On remarque alors que le système complété ne permet que d'obtenir une solution semi-décidable pour résoudre le problème du mot (i.e. permettant de déduire les implications), puisqu'on n'utilise pas d'ordre bien fondé pour orienter les règles du système. Cependant, nous avons tout de même introduit quelques optimisations.

En effet, toutes les règles de réécriture sont normalisées des deux côtés chaque fois qu'une nouvelle règle est ajoutée, ce qui favorise la terminaison. De plus, les règles qui se réduisent à elles-mêmes, comme $a \to a$, sont systématiquement supprimées.

Malgré ces précautions, certains systèmes ne parviennent pas à être complétés (environ 60% des systèmes sont complétés). Dans certains cas, la normalisation semble ne pas terminer ; dans d'autres, le nombre de paires critiques explose. Ainsi, si un seuil prédéfini pour le nombre de pas de normalisation et pour le nombre de paires critiques est atteint, nous abandonnons la complétion.

Il est en réalité important de définir ce seuil correctement. En effet, prendre un seuil trop grand augmente considérablement le temps de calcul inutilement. À l'inverse, prendre un seuil trop petit peut amener à rejeter des normalisations qui allaient finalement terminer. Notre expérimentation indique qu'une limite de 64 pas par normalisation et un maximum de 40 paires critiques simultanées semblent suffisants pour assurer une complétion rapide et correcte.

3.3 Orientation

Jusqu'ici, l'orientation des règles n'était pas prise en compte. Il est possible de remédier à cela et d'espérer obtenir de meilleurs résultats. Une première approche consiste à orienter toutes les règles à chaque étape vers le mot le plus court, s'il existe, ou, à défaut, à ne pas leur attribuer de direction particulière. Avec cette méthode, environ 70% des systèmes sont complétés. On note que ces complétions ne fournissent que des solutions semi-décidables pour les implications, puisque l'ordre utilisé n'est pas bien fondé.

Une amélioration consiste à implémenter l'ordre lexicographique. Cela revient à orienter les règles vers le mot le plus court, s'il existe, ou sinon, vers le mot le plus petit selon l'ordre du dictionnaire. Cette approche permet de compléter environ 80% des systèmes. Pour la première fois, on obtient une solution décidable pour définir les implications.

Cependant, même cet ordre bien connu en réécriture ne permet pas de compléter tous les systèmes. Il devient alors nécessaire d'explorer des ordres spécifiques pour les systèmes restants. Un autre ordre bien fondé permettant de compléter quelques systèmes supplémentaires est l'ordre du dictionnaire, c'est-à-dire orienter vers le mot le plus petit au sens du dictionnaire.

Ensuite, après de nombreux tests, il apparaît que des ordres contre-intuitifs peuvent parfois permettre de terminer la complétion. Par exemple, orienter systématiquement vers le mot le plus grand, soit au sens de l'ordre lexicographique, soit au sens de l'ordre du dictionnaire. Dans certains cas, orienter les règles vers le mot le plus long empêche la formation de nouvelles paires critiques (le système était déjà complet), tandis que dans l'autre sens, la complétion ne parvenait pas à terminer.

Ainsi, aucun ordre ne semble garantir la terminaison de tous les systèmes de réécriture. Une approche pragmatique consiste à tester la complétion avec plusieurs ordres successifs, en espérant qu'un ordre convienne pour chaque système. Avec cette méthode, environ 90% des systèmes sont complétés.

3.4 Générateur

Nous allons à présent examiner des techniques permettant de compléter tous les systèmes contenant des mots de longueur au plus 4 de manière convergente, afin d'obtenir une solution décidable pour déterminer les implications.

3.4.1 Sous-chaînes des mots

Parmi les systèmes restants à ce stade figure l'équation $aba = bab$, bien connue dans le domaine de la réécriture. En effet, il n'existe aucune manière d'orienter cette règle et son système de réécriture pour permettre la terminaison de la complétion.

Cependant, une méthode permet de compléter ce système : utiliser les transformations de Tietze, en particulier l'ajout de générateurs. Dans ce cas précis, il suffit d'ajouter le générateur c et la règle de réécriture $ab = c$. Avec cette modification, la complétion parvient effectivement à son terme.

Malheureusement, il n'existe pas de méthode systématique pour déterminer quel générateur et quelle règle ajouter afin de garantir la terminaison de la complétion. Par conséquent, pour les systèmes de réécriture suivants, nous avons expérimenté en ajoutant un générateur et une règle de manière arbitraire.

Prenons l'exemple mentionné plus haut. Si l'alphabet contient déjà a et b, on introduit c comme nouveau générateur en choisissant simplement le premier disponible selon l'ordre de l'alphabet. Ensuite, nous testons différentes règles. Nous commençons avec le mot aba et essayons successivement les règles suivantes jusqu'à obtenir une complétion qui termine : $a = c$, puis $b = c$, puis $ab = c$, puis $ba = c$, puis $aba = c$. Enfin, si ces essais ne donnent toujours pas de résultat concluant, nous testons avec le côté droit de l'équation, ici bab.

Grâce à cette technique, nous parvenons à compléter 95% des systèmes de réécriture.

3.4.2 Sous-ensembles des sous-chaînes des mots

Nous allons maintenant introduire une nouvelle méthode permettant de compléter davantage de systèmes.

Tout d'abord, on remarque que les transformations de Tietze permettent d'ajouter plusieurs générateurs. En effet, si l'on ajoute un nouveau générateur, le système obtenu reste équivalent au précédent. De la même manière, si l'on ajoute un deuxième générateur, le nouveau système est équivalent au précédent, et ainsi de suite.

On peut alors se demander si ajouter plusieurs générateurs associés à différentes sous-chaînes permettrait de débloquer certaines complétions. Pour ce faire, nous construisons un ensemble E dans lequel chaque sous-chaîne est associée à un nouveau générateur (toujours en choisissant le premier disponible selon l'ordre de l'alphabet). Ensuite, nous formons un nouvel ensemble SE, qui contient tous les sous-ensembles de E.

Il ne reste plus qu'à tester, pour chaque ensemble de SE, si la complétion termine en ajoutant les nouvelles règles correspondantes. Et en effet, la plupart des complétions problématiques aboutissent.

Grâce à cette méthode, 98% des systèmes de réécriture sont à présent complétés.

3.4.3 Expansions des mots

Intéressons-nous maintenant à une dernière technique permettant de compléter les derniers systèmes restants.

On observe d'abord que lors de la complétion des systèmes, la taille des mots apparaissant dans les nouvelles règles semble croître de manière arbitraire. Ainsi, on peut se demander s'il ne serait pas judicieux d'ajouter des générateurs associés à des mots plus longs que ceux déjà présents dans la règle.

Pour cela, nous prenons toutes les sous-chaînes et les ajoutons soit au début, soit à la fin, soit aux deux extrémités des mots de la règle. Par exemple, pour la règle $ab \to ba$, nous testons successivement les règles suivantes : $c = aab$, $c = aba$, $c = bab$, $c = abb$, $c = abab$, $c = ababab$, etc.

L'introduction de ces nouveaux générateurs permet de compléter 99% des systèmes de réécriture (une seule règle reste incomplète). Pour compléter ce dernier cas, nous pouvons appliquer la méthode des sous-ensembles mentionnée précédemment.

Concrètement, nous récupérons tous les sous-ensembles des sous-chaînes (le même ensemble SE que précédemment). Ensuite, pour chaque mot de la règle, nous appliquons une transformation qui consiste à ajouter chaque sous-chaîne au début du mot. Nous testons chaque sous-ensemble tant que le système n'est pas complété.

En cas d'échec, nous répétons l'opération en ajoutant les sous-chaînes à la fin du mot, puis en les ajoutant simultanément au début et à la fin.

Finalement, nous atteignons le cap des 100% de systèmes de réécriture complétés.

3.5 Implications

À présent, nous avons réussi à compléter tous les systèmes de réécriture. Nous pouvons alors examiner quelles équations impliquent quelles autres équations. Pour ce faire, nous prenons une équation ainsi que son système de réécriture complété. Ensuite, pour chaque autre équation, nous normalisons ses deux côtés à l'aide des règles de réécriture du système. Si les deux côtés sont égaux, alors il y a implication. Sinon, l'équation associée au système n'implique pas l'équation en question.

Une fois que nous avons déterminé toutes les implications, nous pouvons générer le diagramme de Hasse. Pour cela, il faut supprimer les implications qui apparaissent en double par transitivité. Il suffit ensuite d'effectuer une exploration récursive des implications de chaque règle pour simplifier correctement les relations.

Enfin, nous pouvons utiliser Graphviz, notamment son outil *Finite Automaton*, pour représenter simplement le diagramme final. Nous générons également un fichier annexe qui liste les équations n'apparaissant pas dans les implications, c'est-à-dire celles qui n'impliquent aucune autre équation et qui ne sont impliquées par aucune équation. Nous générons également d'autres graphes, comme le graphe des règles homogènes et le graphe des règles égales par renversement.

Une observation concernant les diagrammes de Hasse des mots engendrés par l'alphabet $\Sigma = \{a, b\}$: filtrer les mots à symétrie et renommage près suffit pour obtenir un graphe sans cycle, c'est-à-dire un préordre acyclique sur chaque ensemble, ce qui n'était pas évident au départ.

4 Statistiques

4.1 Temps d'exécution

Comme nous considérons toutes les sous-chaînes des deux mots de la règle, puis tous les sous-ensembles de ces sous-chaînes, le temps de calcul est exponentiel en la taille des mots.

Ainsi, si, pour les règles dont les mots sont de longueur au plus 4, le calcul des complétions est quasi-instantané, il n'en est pas de même lorsque les mots sont de longueur au plus 5. En effet, le temps de calcul est d'environ 30 minutes.

De plus, ces temps de calcul sont obtenus avec du parallélisme à 8 cœurs.

4.2 Suites de nombres

Nous avons essayé de reconnaître des suites de nombres connus dans certaines propriétés de nos diagrammes de Hasse. Nous n'avons rien trouvé pour le nombre d'arêtes, le nombre de nœuds internes et le nombre de feuilles. Néanmoins, nous avons trouvé des correspondances pour le nombre de nœuds ne participant pas aux implications.

La suite de nombres pour ces nœuds dans les différents graphes est la suivante : 0, 1, 4, 9, 25. Nous avons trois correspondances :

La première : A055851, $a(n)$ et $\lfloor a(n)/6 \rfloor$ sont tous deux des carrés ; c'est-à-dire des nombres carrés qui restent des carrés lorsqu'ils sont écrits en base 6 et que leur dernier chiffre est supprimé. Le début de la suite : 0, 1, 4, 9, 25, 100, 729, 2401, 9604, 71289.

La deuxième : A238334, nombres carrés ne contenant aucune sous-chaîne plus courte qui soit un carré. Le début de la suite : 0, 1, 4, 9, 25, 36, 576, 676, 5776, 27556.

La troisième : A117678, nombres carrés dont la racine numérique multiplicative est également un carré. Le début de la suite : 0, 1, 4, 9, 25, 100, 169, 196, 225, 256.

5 Conclusion

5.1 Résultats

L'implémentation de la *Complétion de Knuth-Bendix* et les expérimentations sur les ordres de réduction et les générateurs nous ont permis de compléter automatiquement tous les systèmes sur les mots de longueur au plus 4. Ainsi, nous avons pu déterminer toutes les implications engendrées par ces ensembles et générer les diagrammes de Hasse.

Nous avons ainsi pu montrer un premier résultat : tout système de réécriture d'une règle composée de mots de longueur au plus 4 peut être complété de façon convergente.

Nous avons également montré un deuxième résultat : la symétrie et le renommage suffisent pour obtenir un préordre acyclique sur cet ensemble.

5.2 Limitations

Tout d'abord, nos méthodes ne permettent pas de compléter de manière convergente les systèmes pour les mots de longueur au plus 5. En effet, trois règles bloquent : $aab \to baaba$, $abab \to baaba$, $abb \to bbabb$.

De plus, l'utilisation d'ordres non bien fondés permet de compléter ces systèmes. Malheureusement, la normalisation ne termine pas toujours dans le calcul des implications. Néanmoins, on peut donc envisager l'élaboration d'une solution impliquant des ordres bien fondés.

Ensuite, l'étude des systèmes pour les mots de longueur au plus 6 peut commencer à devenir compliquée à cause du temps de calcul. En effet, considérer tous les sous-ensembles de toutes les sous-chaînes est exponentiel. On le remarque bien puisque le temps de calcul pour les mots de longueur au plus 5 était déjà considérablement plus long que pour ceux de longueur 4.

On peut donc légitimement penser qu'il faudrait développer de nouvelles solutions moins coûteuses afin de compléter des systèmes portant sur des mots bien plus longs.

Enfin, nous aurions voulu étudier les graphes des présentations à plusieurs relations. Néanmoins, leur génération est particulièrement complexe. En effet, la génération du fichier *dot* n'est que la partie la plus simple, puisque sa compilation en *pdf* est plus délicate. Par défaut, la compilation semble interminable. Si l'on ajoute des options pour l'accélérer, le résultat devient illisible. Ainsi, cette étude a été laissée en suspens.

5.3 Perspectives

Un objectif à court terme est de terminer l'étude de la *Complétion de Knuth-Bendix* pour les mots de longueur au plus 5 et de commencer celle pour les mots de longueur au plus 6. Un autre objectif est d'étudier plus en détail les diagrammes de Hasse pour les mots de longueur au plus 4, notamment les graphes des présentations à plusieurs relations.

À plus long terme, plusieurs options s'offrent à nous. Bien que le projet de *Terence Tao* soit à présent terminé, nous pourrions essayer d'adapter nos méthodes pour résoudre les implications, puisque la réécriture n'a pas été utilisée dans la résolution du projet. Un autre axe de travail serait d'implémenter la génération automatique des preuves en *Lean*, qui n'a pas encore été étudiée.

Une autre possibilité serait de se pencher sur le problème ouvert de la décidabilité du mot pour les monoïdes mono-relationnés. Pour ce faire, nous pourrions chercher à prouver que l'égalité sur les monoïdes finis est indécidable en encodant une machine de Turing à une seule règle simulant les différentes transitions.

5.4 Remerciements

Je tiens à remercier Samuel Mimram pour son encadrement et son aide tout au long de cette année de projet, qui s'est révélée particulièrement enrichissante.

Références

[NB21] Carl-Fredrik Nyberg-Brodda. The word problem for one-relation monoids : A survey, 2021.

[Tao24] Terence Tao. A pilot project in universal algebra to explore new ways to collaborate and use machine assistance ? Blog, 2024.

Annexes

Paires critiques

Dans cette section, nous expliquons en détail le processus d'obtention des paires critiques.

Les paires critiques sont calculées à partir de deux règles, r_1 et r_2. Pour chaque règle, il est nécessaire d'extraire le mot qui peut être réécrit en un autre. Par exemple, pour la règle $a \to b$, on extrait a. Ces mots seront notés w_1 et w_2.

Ensuite, nous superposons w_1 sur w_2 afin d'identifier tous les mots qui se superposent correctement. Dans l'exemple suivant, le caractère _ représente une position vide. Par exemple, avec les règles $r_1 : ab \to c$ et $r_2 : ba \to d$, nous essayons de superposer $ab\_$ sur $\_ba$. Cette superposition est correcte, car a et _ sont compatibles, b et b sont égaux, et _ et a sont compatibles. Ainsi, nous obtenons le mot aba. En revanche, superposer ab sur ba ne donne pas une correspondance valide. Ensuite, nous superposons $\_ab$ sur $ba\_$, ce qui est une superposition correcte et nous donne le mot bab.

Pour chaque mot obtenu, nous calculons toutes les formes normales en utilisant les deux règles r_1 et r_2. Les ensembles des formes normales obtenues sont notés n_1 et n_2. Les paires critiques sont ensuite calculées de la manière suivante : $\{\forall fn_1 \in n_1, \forall fn_2 \in n_2, (fn_1,\ fn_2)\}$. En reprenant l'exemple précédent, aba se réécrit en ca avec r_1 et en ad avec r_2, tandis que bab se réécrit en bc avec r_1 et en db avec r_2. Nous obtenons alors les paires critiques (ca, ad) et (bc, db).

Complétion de Knuth-Bendix

Dans cette section, nous allons expliquer plus en détail le fonctionnement de la *Complétion de Knuth-Bendix*.

La *Complétion de Knuth-Bendix* est calculée à partir d'un système de réécriture et d'un ordre de réduction. La première étape consiste à orienter toutes les règles du système en fonction de l'ordre de réduction donné. Ensuite, on ajoute dans une queue toutes les règles du système de réécriture.

Tant qu'il reste des règles dans la queue, on prend la première règle (et on la retire de celle-ci), puis on calcule toutes ses paires critiques avec l'ensemble des règles du système (y compris elle-même). Ensuite, on ajoute toutes les nouvelles règles obtenues, à partir des paires critiques, au système de réécriture et à la queue. Une règle est obtenue à partir d'une paire critique (w_1, w_2) en normalisant w_1 et w_2, puis en orientant la règle $w_1 \to w_2$ en fonction de l'ordre de réduction donné pour la complétion.