Utiliser, Comprendre et Modifier le Prépabot

Comité PrépaBrinks (Iago)
La Cagette de Montpellier
Dernière mise à jour: 2026-04-06

Contexte

La Brinks est le comité Cagette qui s'occupe de vérifier, semaine après semaine, que les paiements enregistrés sur l'ordinateur de caisse au magasin correspondent bien aux paiements effectivement reçus dans la caisse, à la banque, en MonA *etc.* Parce que parfois, en effet, ça ne correspond pas. Les humain·es, comme les machines, font des petites erreurs.

La Brinks s'occupe aussi d'amener les espèces à la banques, de contacter les coops en cas d'erreur *etc.* mais le sujet de ce document c'est surtout la *vérification*.

La vérification

L'enjeu de cette vérification c'est de pouvoir rappeler un·e coop pour lui rendre son argent quand on s'aperçoit après coup qu'iel a trop payé, ou pour lui réclamer son argent quand au contraire on s'aperçoit qu'on n'a pas reçu assez. Depuis mai dernier, par exemple, c'est environ 3 400€ d'erreurs qui ont été rendus/récupérés/corrigés de cette manière. La vérification permet aussi de fournir des fichiers corrects aux services comptables de la Cagette en aval, avec des totaux pour l'exercice de chaque mois/année/*etc.* qui tombent bien juste.

La Brinks se réunit chaque semaine pour comparer deux "flux" de données collectés au cours de la semaine précédente :

- Un flux de transactions **attendues**, enregistré par notre logiciel de caisse **Odoo** chaque fois qu'un·e coop passe en caisse.
- Un flux de transactions effectivement **reçues**, consulté par les brinkseur·ses.

Les deux flux doivent correspondre. S'il ne correspondent pas c'est qu'il y a des **anomalies** à mettre en valeur, et des décisions à prendre pour les résoudre.

C'est assez simple à comprendre, mais plutôt délicat à mettre en œuvre pour diverses raisons qu'on va toutes expliquer dans ce document. La première, c'est qu'il existe de nombreux moyens de paiement acceptés à la Cagette.

Les moyens de paiement

Quand on passe à la caisse, on peut payer :

- En **espèces** : Odoo enregistre une transaction attendue "en espèces", et le liquide se retrouve dans la caisse. Comme il s'y mélange avec toutes les autres, on n'a pas de *"flux de transaction reçues"* bien défini pour cette modalité. Tout ce qu'on a c'est un **total reçu** calculé pour chaque **session** caisse. C'est-à-dire un total chaque fois qu'un·e coop quitte la caisse pour être remplacé par un·e autre ou bien pour fermer le magasin. On espère que le total reçu corresponde au total attendu, sinon il faut chercher des anomalies.
- Par **chèque** : Odoo enregistre une transaction attendue "par chèque", et le chèque se retrouve dans la caisse. Comme chaque chèque est bien distinct des autres avec un montant précis, on a bien un "flux de chèques reçus" qui est plus facile à vérifier en regard du flux attendu.
- En "ticket restaurant" = **chèque déjeuner** : Odoo enregistre une transaction attendue "en chèque déjeuner" et le·s chèque·s se retrouve·nt dans la caisse. Le montant d'un chèque déjeuner est déjà imprimé sur le coupon, donc ça fonctionne comme des espèces sauf qu'on ne rend pas la monnaie dessus. En particulier, comme avec les espèces, on n'a pas de "flux de transactions attendues" clair à vérifier, et on doit se contenter de comparer la somme de tous les coupons reçus avec la somme attendue.

Les trois moyens de paiement ci-dessus sont *physiques*, avec des coupons / coupures / pièces / chèques qui se trouvent effectivement dans la caisse. C'est pendant la réunion hebdomaire de la brinks que tous ces objets reçus sont comptés, vérifiés, puis déposés à la banque.
Mais la Cagette accepte encore d'autres modalités. Quand on passe à la caisse, on peut aussi payer :

- En **MonA** : Odoo enregistre une transaction attendue "en mona", puis on se connecte avec la tablette pour effectuer la transaction sur le site de la MonA avec notre compte client. Plus tard, en se connectant au site avec le compte de la Cagette, la Brinks peut facilement obtenir son "flux de transactions reçues" sous la forme d'un tableur, à comparer avec l'attendu. L'argent se trouve alors "chez la MonA". De temps en temps, la Cagette réclame un gros virement auprès de la mona pour le rapatrier sur le compte bancaire de la Cagette, mais ce n'est pas la Brinks qui s'en occupe.
- En carte bancaire = **CB** : Odoo enregistre une transaction attendue "en CB", puis on passe notre carte sur le TPE (pour le sans-contact) ou bien on l'insère et on tape le code (pour le

reste). À la fin de chaque session caisse, les deux totaux “contact” et “sans-contact” enregistrés par le TPE sont envoyés à la Banque Populaire = **bpop**. Plus tard, en se connectant au site bpop avec le compte Cagette, la Brinks peut ~~facilement~~ obtenir son “flux de transaction reçues” sous la forme d’un tableur, à comparer avec l’attendu. (Ce n’est pas du tout facile mais c’est un autre sujet.)

- En “carte déjeuner” = **CBdèj'** : Il y a au moins 5 entreprises différentes = **vendeurs** qui fournissent ces cartes: Pluxee, Edenred, Swile, Up, Bimpli. Quand on les utilise pour payer en caisse, Odoo enregistre une transaction attendue “en CBdèj'”, et y adjoint le nom du vendeur. Ensuite on paie avec la CBdèj' en utilisant le TPE comme pour une CB. Plus tard, pour obtenir ses “flux de transactions reçues” sous la forme de tableurs, la Brinks doit se connecter au site de chacun des vendeurs avec le compte Cagette.
  Par ailleurs, il existe deux différents types de transactions CBdèj' :
  - **Différé** : ces transactions sont enregistrées et accumulées sur le site vendeur. Comme pour la MonA, l’argent se trouve chez eux et on doit le vérifier. De temps en temps, la Cagette va réclamer un gros virement auprès du vendeur pour rapatrier l’argent chez bpop mais ce n’est pas la Brinks qui s’en occupe.
  - **Direct** : ces transactions sont aussi enregistrées sur le site du vendeur, mais elles sont directement reçues chez bpop comme s’il s’agissait de transactions CB. Or, comme elles sont inattendues en CB, elles apparaissent d’abord comme des *anomalies CB*. La Brinks doit faire le rapprochement en comprenant qu’il ne s’agit pas de reçus CB inexpliqués, mais de transactions CBdèj' directes.

  Comme on dispose du détail des transactions, on peut les identifier par leur montant. Mais il est impossible de payer plus de 25€ à la fois en CBdèj'. Le résultat c’est que de nombreuses transactions CBdèj' font exactement 25€ et ça rend l’identification difficile. Si on attendait 15 transactions de 25€ en CBdèj' mais qu’on n’en a reçu que 13, c’est difficile de savoir quel·les coops recontacter, il faut faire une enquête.

Les trois moyens de paiement ci-dessus sont *dématérialisés*, avec des cartes, des comptes, des sites web, *etc.*

La PrépaBrinks

Vérifier les flux dématérialisés c’est trop de travail pour la Brinks, qui doit déjà vérifier les flux physiques chaque semaine. Il existe donc un “comité adjoint”, la *PrépaBrinks*, qui se charge de vérifier chaque semaine tous les flux dématérialisés, puis de les résumer avec les anomalies détectées dans un seul fichier tableur. Ça permet à la Brinks de terminer plus facilement son travail.

Le travail de la PrépaBrinks est donc de produire, chaque semaine, ce fichier “de préparation” contenant tous les flux attendus, tous les flux reçus dématérialisés, et toutes les anomalies visibles sur les flux dématérialisés.

Ce fichier contient une feuille par jour avec un résumé des totaux attendus par session caisse, une section par moyen de paiement avec les transactions attendues et dématérialisées reçues, et un cartouche pour résumer les anomalies repérées. Pour les transactions dématérialisées, la somme des attendus doit égaler celle des reçues + celles des anomalies.

C'est un travail plutôt fastidieux parce qu'il faut:

- (1) Se connecter à tous les différents sites pour obtenir les flux (Odoo, MonA, bpop, vendeurs CBdèj').
- (2) Les convertir chacun depuis des formats éclectiques dans un seul format "prépa".
- (3) Pointer soigneusement chaque transaction dématérialisée en reportant les anomalies.

On y passe des heures, ce n'est pas très rigolo, alors on essaie aussi d'écrire des programmes dans l'espoir que les machines nous aident un peu.

Le `prépabot`

Ce document introduit le programme `prépabot` disponible à l'adresse https://cleb.io/prepabrinks, qui a pour ambition d'automatiser (2) et (3),

L'utiliser

On suppose ici que vous êtes déjà un·e (Prépa)Brinkseur·se

Pour utiliser le prépabot, choisissez le numéro de semaine à préparer, puis récupérez tous les fichiers d'export usuels qui correspondent à cette semaine sur les différents sites (1). Vous pouvez vous aider pour cela de l'autre robot `prepaprepa`, et c'est d'ailleurs *nécessaire* pour les exports bpop qui n'existeraient pas autrement. Ne transformez pas ces fichiers et donnez-les tels quels au prépabot.

Cliquez sur le bouton principal du bot. Ça devrait être assez rapide. Si tous se passe bien, trois fichiers sont produits, que vous pourrez télécharger :

- `archive.zip`: Si le robot a bien *reçu* vos fichiers, il vous propose cette copie de tout ce que vous lui avez donné. C'est utile pour archivage ou bien pour enquêter en cas de problème.
- `data.pc`: Si le robot a bien *compris* vos fichiers, il vous propose cette représentation interne de tous ses "flux attendus" et "flux reçus". Vous ne pourrez pas l'ouvrir vous-même, mais c'est utile pour enquêter en cas de problème.
- `prepa.ods`: Si tout s'est bien passé par la suite, vous obtiendrez ce tableur qui contient la préparation automatique de la semaine. Il vous appartient de la consulter, de la vérifier puis de l'améliorer si nécessaire.

Les interruptions

Le robot est conçu pour être assez *rigide*, et il va préférer s'interrompre dès qu'il remarque quelque chose d'inattendu en essayant de vous expliquer ce qui se passe, plutôt que de continuer comme si de rien n'était au risque de produire une prépa erronée. Dans ce cas:

- Si vous pensez qu'il vous interrompt pour une bonne raison, ou pour quelque chose de pas très grave, vous pouvez corriger à la main les fichiers (1) jusqu'à ce qu'il se rassure.
- Si vous pensez que tout va bien et qu'il ne devrait pas vous interrompre, alors il faut peut-être l'améliorer. Au choix :
  - Reportez le problème au comité PrépaBrinks, en espérant qu'on puisse s'en occuper assez vite.

- Transformez le robot vous-même. Le code est disponible en ligne à cette adresse. On sait que ça prend du temps et que ça demande de bien comprendre comment il fonctionne. C'est le sujet de la suite du document.
- Laissez tomber le robot et préparez le fichier vous-même à l'ancienne. Ça vous prendra (bien) plus de temps, mais ça fait parfois, euh, du bien à l'humain de se passer des machines ? En fait surtout : on est désolé·es pardon on a fait du mieux qu'on a pu ça reste un robot artisanal :(

Le fichier produit

Le fichier de prépa automatique est conçu pour (beaucoup) ressembler au fichier de prépa traditionnel, mais il contient plus d'information. Voici ce que vous y trouverez de nouveau :

- **Session**: Toutes les transactions attendues, récupérées chez odoo, sont associée à l'*heure* et la *session caisse* auxquelles elles ont été enregistrées. Par exemple si vous lisez `"19h35 c4s5"`, c'est qu'il s'agit d'une transaction enregistrée par Odoo à 19h35 pendant la 5ème session de la caisse numéro 4 ce jour-là.
- **Anomalies**: Les anomalies détectées par le prépabot sont listées dans le cartouche rouge habituel et préfixées par la petite icône 🤖. Elles sont assez verbeuses parce qu'il préfère donner un maximum d'information sans se permettre de trop extrapoler parce que c'est une machine. N'hésitez pas à les simplifier si vous le jugez nécessaire.
- **Deltas**: Les montants associés aux anomalies sont comptés, comme d'habitude, positivement en faveur de la Cagette et négativement en sa défaveur, de sorte que le "delta final" en bas des cartouches vaille exactement zéro pour la MonA et les CB. Si ce n'est pas le cas, c'est un bug du robot à reporter au plus vite. Le delta pour les CBdèj' est plus difficile à interpréter à cause de l'existence des transactions directes, et il est normal qu'il ne vale pas toujours zéro.
- **Commentaires**: Chaque transaction, attendue ou reçue, fait parfois l'objet d'un commentaire du robot, préfixé par l'icône 🤖 à moins qu'il ne soit très court. Ce commentaire est supposé vous aider à faire le lien entre les transactions connectées. S'il n'est pas utile, ou s'il prête à confusion, n'hésitez pas à le reporter.
- **Paquets**: Le robot travaille en établissant des connections entre les transactions, par exemple quand les montants sont identiques. Chaque jour de la semaine, les paquets qui contiennent 3 transactions connectées ou plus sont automatiquement nommés par des lettres minuscules comme `[a]` ou `[bc]`. Ces annotations vous permettent de repérer ces paquets, qui peuvent parfois s'étaler sur plusieurs moyens de paiement différents. Par exemple, si quelqu'un·e paie exactement 13€44 en MonA et qu'on reçoit aussi 13€44 de la part de quelqu'un·e d'autre chez Swile, la lettre du paquet vous indique la connection entre ces transactions pour vous permettre de les retrouver facilement.
- **Couleurs**: Le robot va laisser en **blanc** toutes les transactions qui ont été pointées avec quasi-certitude. Les anomalies usuelles seront coloriées en **rouge** : paiement manquant ou paiement inattendu. Les paquets de transactions incertains mais sans conséquence sur le delta sont coloriés en **bleu**. Par exemple si on attend 4 paiments CB de 13€44 et qu'on en

a bien reçu 4, le robot ne reporte pas d'anomalie mais comme on ne sait pas qui est qui il colorie quand même le paquet en bleu pour prévenir que ce n'est pas clair. Enfin, si ce n'est pas clair et que ça impacte le delta, il colorie le paquet en **violet**. Par exemple, si on n'a reçu que 3 paiements CB de 13€44 dans l'exemple ci-dessus, on sait qu'il en manque 1 mais on ne sait pas qui. Le brinkseur ou la brinkseuse est alors en charge d'enquêter pour comprendre. En d'autre termes: *le **violet** appelle une intervention humaine: le robot est certain de ne pas pouvoir comprendre exactement ce qui s'est passé.*

- **Ordre**: Les transactions sont principalement rangées par ordre chronologique. Quand le site ne donne pas l'heure de la transaction, elles sont rangées par ordre d'apparition dans l'export original. Le robot se permet cependant de rompre brièvement l'ordre pour regrouper des transactions qu'il juge connectées ensembles, et pour faciliter la lecture.
- **CB**: Comme elles sont très nombreuses, les transactions CB sont reportées dans une feuille séparée pour chaque jour de la semaine. Elles y sont regroupées par session caisse, en y associant les remises que le robot a automatiquement devinées. Chaque remise bpop y est identifée par une lettre majuscule comme `A` ou `B`.
- **CBdèj'**: Trois nouvelles colonnes apparaissent dans la section CBdèj'. Celles de gauche reportent le vendeur chez qui les transactions reçues ont été trouvées et leur montant. Celle du milieu explique le statut de la transaction:
  - : (rien) pour une transaction *différée*.
  - `[ ]`: pour une transaction *directe* qu'on n'a *pas* retrouvée chez bpop.
  - `[x]`: pour une transaction *directe* qu'on a *bien* retrouvée chez bpop.
  - `[?]`: pour une transaction *directe* dont le robot *ne sait pas* si elle est arrivée bpop.
  - `[!]`: pour une transaction CBdèj' *inattendue* qui se trouve être *directe*.
  - `[o]`: pour une transaction CBdèj' *directe*, *inattendue*, mais *pas* trouvée chez bpop (donc sans incidence sur le bilan).

  Pas d'inquiétude, les commentaires et les anomalies vous réexpliqueront tout ça.

Le robot est conçu pour durer dans le temps, s'améliorer, et répondre aux besoins changeants de la Cagette. Ça demande du travail, et ça nécessite qu'on sache s'il ne fonctionne pas de façon satisfaisante. N'hésitez pas à reporter le moindre problème au comité.

Comment ça marche

Cette section est écrite à l'attention des (Prépa)Brinkseur·ses qui souhaiteraient *comprendre* comment le `prépabot` fonctionne, et elle contient aussi des informations utiles pour celles et ceux qui souhaiteraient *modifier* son fonctionnement. Par exemple, on pourrait vouloir modifier le robot parce qu'on a trouvé un bug, parce qu'on voudrait améliorer son verbiage, parce que le format des fichiers d'entrée a changé sur les sites, parce qu'on aimerait modifier le format des fichiers de sortie, ou parce que les modalités de paiement à la Cagette ont évolué. Si vous ne voulez rien de tout ça mais que vous êtes quand même curieux·se, bienvenue dans la section quoi qu'il en soit :)

La technologie

Le travail de la PrépaBrinks est plutôt simple dans son ensemble et plutôt horriblement compliqué dans ses détails. En conséquence, le projet de développement qui devait conduire au `prépabot` est devenu plutôt sérieux. Le programme est écrit en Rust, un langage de programmation moderne et open-source, et comporte aujourd'hui presque 10 000 lignes. En conscience de cette complexité, une attention importante a été apportée à la qualité du code et à sa documentation (dont ce document fait partie), avec une mise en valeur particulière des parties les plus susceptibles de devoir changer dans le futur : format des fichiers d'entrée, nombre de vendeurs CBdèj', détail des annotations..

Le langage Rust a de nombreuses qualités, mais c'est principalement pour sa *rigueur* qu'il a été choisi ici, parce que la préparation Brinks est un problème plutôt compliqué auprès duquel il est facile de se mélanger les pinceaux. Écrire le robot en Rust rend plutôt difficile l'introduction d'erreurs dans le programme (futiles comme très graves), même par des intervenant·es futur·es. Le style de programmation "défensive" choisi met l'accent sur la prévention de ces erreurs. Si vous devez modifier le code et qu'il faut choisir entre sûreté et simplicité, choisissez la sûreté. S'il faut choisir entre lever une erreur ou ignorer un anomalie, choisissez de lever une erreur.

Le code est hébergé en ligne chez https://codeberg.org/iago-lito/prepabrinks, une forge logicielle open-source européenne détenue par une fondation allemande non-lucrative. Les collaborations y sont bienvenues sous la forme de PR, de bug reports ou de simples discussions Le code est distribué sous la licence libre GPL-3.

L'application web est constituée de (très) simples fichiers HTML/CSS/JavaScript, accompagnés des binaires WASM produits par la compilation du robot. Elle est hébergée en ligne sur mon serveur personnel (Iago) à l'adresse https://cleb.io/prepabrinks, mais elle est entièrement *statique*. Ça veut dire que le robot tourne dans votre propre navigateur, sur votre propre ordinateur, et ça a des conséquences importantes pour la Cagette :

- Les fichiers déposés et produits restent sur votre ordinateur, personne ne peut en prendre connaissance même si vous les donnez au robot.
- Vous pouvez travailler hors-ligne (à condition d'avoir téléchargé l'application).
- Vous pouvez télécharger, modifier le code et obtenir votre propre version pour vous en convaincre.

La (présente) documentation est écrite avec Typst, un langage de composition moderne et open-source, et son code distribué au sein du même projet, à la même adresse et sous la même licence.

Les entrées

Les fichiers d'entrée sont des fichiers `.xlsx` ou bien `.csv`. Comme ils ont des contenus plutôt variables, ils sont analysés par une partie du code qui anticipe des changements fréquents en s'abstrayant du format sous-jacent.

Pour mettre à jour les entrées, il faut modifier les fichiers contenus dans le dossier `src/load/`, on y trouve le nom et la position des colonnes attendues, ainsi les vérifications qui sont faites sur chaque valeur trouvée.

Les évènements

Toutes les données lues dans les fichiers d'entrées sont d'abord aggrégées par le robot en un seul immense flux d'évènements ordonnés "à plat", décrit dans le module `src/events.rs`. Pour s'en faire une idée, voici des exemples d'évènements vus par le robot:

- Ouverture d'une session caisse.
- Enregistrement d'un paiement attendu en CB.
- Enregistrement d'un paiement attendu en Espèces.
- Réception d'un paiement chez Swile.
- Enregistrement d'un paiement attendu en chèque.
- Ouverture d'une autre session caisse.
- Réception d'un paiement en MonA.
- Enregistrement d'un paiement attendu en CB.
- Fermeture d'une session caisse.
- *etc.*

Évidemment on ne peut pas ordonner tous les évènements parce que certains sites ne nous donnent que leur date et pas l'heure. Les évènements trouvés sur ces sites-là sont collectés dans un "sous-flux" à part, moins précis mais conceptuellement identique.

Les remises bpop en particulier (appelées `remittances` dans le code), sont particulièrement gauches parce qu'elles sont datées d'un jour qui ne correspond pas forcément aux facturettes (`receipts`) qu'elles contiennent. Typiquement par exemple, les facturettes du samedi nous parviennent dans des remises datées du lundi, mais il arrive aussi que les remises après un jour férié ou un pont contiennent un mélange de facturettes issues de différentes dates. Fort heureusement les facturettes sont aussi datées, ce qui permet au robot d'associer chaque remise avec l'ensemble des *vraies* dates qu'elle contient et de retrouver ses petits par la suite.

C'est l'ensemble de ces données qui constitue la base de travail du robot, sa représentation interne, et qui devient disponible dans le fichier `data.pc` quand il a terminé de travailler. À ce stade, aucun raisonnement n'a encore été effectué.

L'extraction

Pour chaque journée de la semaine à préparer, le robot extrait du flux unique les évènements datés du jour, et raisonne en isolation sur cet extrait. (Ça veut dire qu'un paiement attendu le mardi et reçu le mercredi sera vu comme deux anomalies par le robot mais zéro par le·la brinkseur·se.)

Il commence par étudier les évènements d'ouverture/fermeture de session caisse (`desk` dans le code). en vérifiant que chaque caisse ouverte est bien fermée plus tard *etc.* Ça lui permet ensuite d'associer chaque transaction attendue à la session caisse correspondante, et de vérifier que les horaires concordent.

Puis il extrait toutes les transactions effectivement reçues pendant ce jour. Pour une remise bpop qui contient plusieurs dates de facturettes différentes, seules les facturettes pertinentes sont extraites.

À ce stade, il n'y a plus de travail à réaliser pour les flux physiques: c'est la Brinks qui s'en occupera lors de sa prochaine réunion. Quant aux flux dématérialisés, ils sont toujours aggrégés, mais on peut conceptuellement les séparer en 7 flux de transactions différentes:

- Les transactions MonA attendues du jour: `mona_expected` = `MNE`.
- Les transactions MonA reçue ce jour: `mona_actual` = `MNA`.
- Les transactions CB attendues du jour: `credit_card_expected` = `CCE`.
- Les transactions CB reçues ce jour: `credit_card_actual` = `CCA`.
- Les transactions CBdèj' attendues ce jour: `meal_card_expected` = `MCE`.
- Les transactions CBdèj' *différées* reçues ce jour: `meal_card_actual_delay` = `MCA_L` = `L`.
- Les transactions CBdèj' *directes* reçues ce jour: `meal_card_actual_direct` = `MCA_D` = `D`.

C'est sur ces flux-là que la partie la plus délicate du travail commence.

La mise en correspondance

Le robot s'attèle alors (dans les modules `src/map`) à mettre en correspondance les transactions entre les flux *attendus* (`expected` = `E`) et *reçus* (`actual` = `A`). Il le fait d'abord en isolation pour les trois moyens de paiement, sans établir de lien entre des paiements de nature différente.

De façon importante, il *désordonne* les transactions pour les mettre en correspondance sans se soucier de leur enchaînement. Ça veut dire qu'un paiement MonA attendu le tôt le matin et reçu tard le soir ne doit pas lui poser problème.

Ensuite pour chaque modalité, il place les paiements attendus "à gauche" et les paiements reçus "à droite", puis il construit un graphe biparti en dessinant un lien (`link`) entre toutes les transactions qui pourraient être en rapport. Pendant cette étape, il s'efforce d'établir *le plus de liens possibles*, même tirés par les cheveux.

Pour la MonA par exemple, un lien est établi si:
- Les montants sont les mêmes.
- Les montants sont les mêmes à un chiffre près.
- Les montants sont les mêmes mais les chiffres sont permutés.
- Les montants sont les mêmes à quelques centimes près.
- Le nom de coop et le nom de membres MonA se ressemblent.

Pour les CBdèj' c'est la même histoire, sauf qu'on n'a pas de nom côté vendeur donc il ne se fonde que sur les montants.

Pour les CB c'est la même chose, mais on peut aussi mettre en correspondance chaque remise avec la session caisse correspondante. On sait que des remises (souvent deux mais parfois plus ou bien une seule) sont créées chez bpop à chaque fermeture de session caisse, mais on ne nous dit pas à quelle heure et elles nous arrivent mélangées. Pour chaque remise du jour et pour chaque session du jour, le robot compte combien de montants sont exactement identiques. Une fois qu'il a testé toutes les combinaisons, il associe d'abord ensemble la session et la remise qui ont le plus de montants en commun, puis il fait la meilleure association parmi celles qui restent *etc.* jusqu'à avoir deviné quelles remises correspondent à chaque session.

L'élagage

Les graphes obtenus à l'étape précédente sont possiblement de véritables sacs de nœuds, qui contiennent beaucoup trop de connexions entre des transactions qui n'ont vraisemblablement rien à voir.

Dans une seconde étape, le robot détermine toute les composantes connexes de ces graphes (`components`). Une composante connexe c'est un ensemble de transactions qui sont toutes connectées les unes aux autres mais pas au reste. Autrement dit si on tire une seule transaction dans la composante alors toute la composante vient avec mais pas le reste. Chaque composante est ensuite étudiée séparément :

- Une composante avec une seule transaction "attendue" est conservée comme une *anomalie négative*. Elle ne contient aucun lien. Le robot n'a rien trouvé qui lui correspond, même en tirant par les cheveux: c'est un paiement manquant : `Missing`. On le garde tel quel.
- Une composante avec une seule transaction "reçue" est conservée comme une *anomalie positive*. Elle ne contient aucun lien. Le robot n'a rien trouvé qui lui correspond, même en tirant par les cheveux: c'est un paiement inattendu : `Unexpected`. On le garde tel quel.
- Une composante avec seulement deux transactions, une "attendue" et une "reçue", est conservée comme une *non-anomalie*. Elle contient forcément un seul lien, et le robot n'a rien trouvé d'autre qui lui correspond même en tirant par les cheveux. C'est un paiement apparié : `Matching`. Si tout se passe bien c'est le cas le plus fréquent. On le garde tel quel.
- Une composante avec plus de deux transactions, c'est potentiellement une espèce de pelotte avec plusieurs liens : `MultiEdge`. Elle pourrait s'avérer difficile à détricoter et il faut essayer de l'améliorer.

Chaque composante `MultiEdge` est donc ré-évaluée pour **élaguer** (`prune`) certaines de ses connexions, dans l'espoir de la séparer en plusieurs nouvelles composantes plus simples.

Par exemple s'il existe, au sein d'une composante, un lien reliant deux transactions qui font *exactement* le même montant, alors tous les autres liens "approximatifs" de la composante sont élagués. Le robot obtient de nouvelles composantes plus petites et dites *uniformes* parce que toutes les transactions qu'elles contiennent partagent exactement le même montant.

Dans une composante MonA, le robot peut aller plus loin : s'il existe un lien reliant deux transactions associées à des noms de coop et de MonA qui correspondent, alors tous les liens dont les noms ne correspondent pas sont aussi élagués.

Dans une composante CB, il y a des liens "intra-session", qui connectent des transactions attendue et reçue pendant la même session caisse, et des liens "inter-sessions", qui connectent des transactions issues de sessions différentes. Si l'élagage des liens inter-sessions résulte uniquement en des composantes intra-session *équilibrées* (mêmes totaux attendu et reçu), alors les liens inter-sessions sont élagués.

Dans une composante CBdèj', de la même façon, il y a des liens "intra-vendeur" et "inter-vendeurs". Si la composante est uniforme (typiquement 25€), alors le robot élague les liens pour former des sous-composantes équilibrées *par vendeur*, et regroupe toutes les transactions

restantes dans une composante “poubelle” (`dump`) qui reste quand même plus petite et plus facile à analyser par le·la brinkseur·se.

L’élagage est répété en boucle jusqu’à ce qu’aucune composante ne puisse plus être réduite. À ce stade, le travail individuel sur chaque modalité est terminé.

Le croisement

Comme il y a parfois des erreurs en caisse, il arrive que des transactions attendues en MonA soient finalement trouvées en CB, ou en CBdèj' *etc.* Par ailleurs, les transactions CBdèj' *directes* sont systématiquement attendues comme des “CB inattendues”.

Pour étudier ces connections-là, le robot doit terminer par une étude inter-modalités. Pour éviter de rendre cette étude trop compliquée, seules les transactions qui font *exactement le même montant* sont susceptibles d’être rapprochées entre deux modalités différentes. (On n’essaie plus de tirer par les cheveux.)

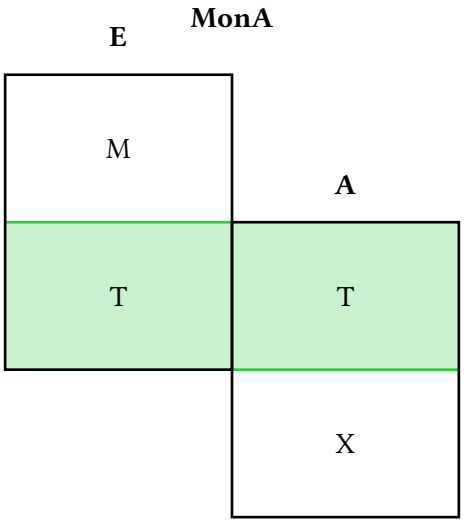
Dans ce contexte, les propriétés individuelles des liens au sein de chaque composante n’ont plus vraiment d’importance. Chaque composante uniforme, quelle que soit sa modalité d’origine (MonA, CB ou CBdèj'), rejoint alors les autres composantes uniformes de même montant dans un **paquet** désordonné de transactions (`bag` dans le code). On pourrait se dire qu’il est invraisembable de rencontrer de gros paquets, parce qu’il est invraisemblable que de nombreuses transactions, un même jour donné, partagent exactement le même montant au centime près. Mais cela se produit *quotidiennement* parce que les CBdèj' sont plafonnées à 25€ et que ce montant précis se retrouve partout dans les fichiers. Le robot doit donc pouvoir étudier de gros paquets.

Comme on ne peut pas systématiquement compter sur le nom, l’heure ou même l’ordre des paiements reçus, le robot n’essaie pas d’utiliser ces informations pour faire correspondre les transactions au sein d’un paquet, et ce travail d’enquête est laissé, si nécessaire, au brinkseur ou à la brinkseuse.

À la place, le robot raisonne simplement sur les *nombres* de transactions identiques trouvées dans les données. Pour rappel, ça fait 7 nombres différents par paquets :

- Les transactions MonA attendues du jour: `mona_expected` = `MNE`.
- Les transactions MonA reçue ce jour: `mona_actual` = `MNA`.
- Les transactions CB attendues du jour: `credit_card_expected` = `CCE`.
- Les transactions CB reçues ce jour: `credit_card_actual` = `CCA`.
- Les transactions CBdèj' attendues ce jour: `meal_card_expected` = `MCE`.
- Les transactions CBdèj' *différées* reçues ce jour: `meal_card_actual_delay` = `MCA_L` = `L`.
- Les transactions CBdèj' *directes* reçues ce jour: `meal_card_actual_direct` = `MCA_D` = `D`.

Voici le principe du raisonnement sur un exemple simple, en imaginant d'abord qu'il n'y aurait que la MonA :



Le bloc de gauche `E` représente l'ensemble des transactions MonA *attendues* de même montant. Le bloc de droite `A` représente l'ensemble des transactions MonA *reçues* de même montant. Si on s'efforce de les faire correspondre, alors toutes les transactions se répartissent en seulement trois catégories:

- Les transactions attendues manquantes : `Missing` = `M`.
- Les transactions correspondantes : `Matched` = `T`.
- Les transactions reçues inattendues : `Unexpected` = `X`.

Tous ces nombres de transactions vérifient les propriétés suivantes :

$$E = M + T$$
$$A = T + X$$

Le robot connaît `E` et `A`, mais il ne peut pas deviner `M`, `T` et `X`. Ce qu'il peut faire, c'est supposer que tout ce qui peut correspondre correspond, et tâcher de *maximiser* T. Ça donne un résultat très simple dans ce cas:

Si $E < A$:	Si $E = A$:	Si $E > A$:
$M = 0$	$M = 0$	$M = E - A$
$T = E$	$T = E = A$	$T = A$
$X = A - E$	$X = 0$	$X = 0$

Autrement dit, soit il n'y a pas de transaction manquante, soit il n'y a pas de transaction inattendue. Il y a toujours un coté qui correspond entièrement à l'autre, et idéalement les deux correspondent exactement quand la situation est parfaite.

On peut aussi voir les trois catégories apparaître dans un tableau comme le suivant:

		0€ ✕	+€ **A**
0€ ✕			X +€
-€	**E**	M -€	T 0€

En ligne les transaction attendues : s'il n'y en a pas on compte 0€ pour la Cagette, s'il y en a on le compte *négativement*. En colonne les transactions reçues : s'il n'y en a pas on compte 0€, s'il y en a on le compte *positivement*. Une transaction qui est à la fois attendue et reçue (en blanc) annule le bilan pour la Cagette (et c'est tant mieux). Une transaction qui n'est pas attendue ni reçue (en gris) ne nous intéresse pas. C'est ce qui fait qu'il n'y a pas $2 \times 2 = 4$ catégories mais seulement 3.

Ce qui rend les choses compliquées, c'est qu'en croisant toutes les modalités dématérialisées il n'y a pas 3 mais $6 \times 4 - 1 = 23$ catégories de transactions :

record \ receipt expected \ actual **E \ A**	0€ ✕	+€ **MonA**	**MealCard**			+€ **Credit-Card**
			+€ **L** *delay*	**D** *direct*		
				0€ *lost* ✕	+€ *passed* **Credit-Card**	
0€ ✕		a +€	b +€	c 0€	d +€	e +€
-€ **MonA**	f -€	g 0€	h 0€	i -€	j 0€	k 0€
-€ **MealCard**	l -€	m 0€	n 0€	o -€	p 0€	q 0€
-€ **CreditCard**	r -€	s 0€	t 0€	u -€	v 0€	w 0€

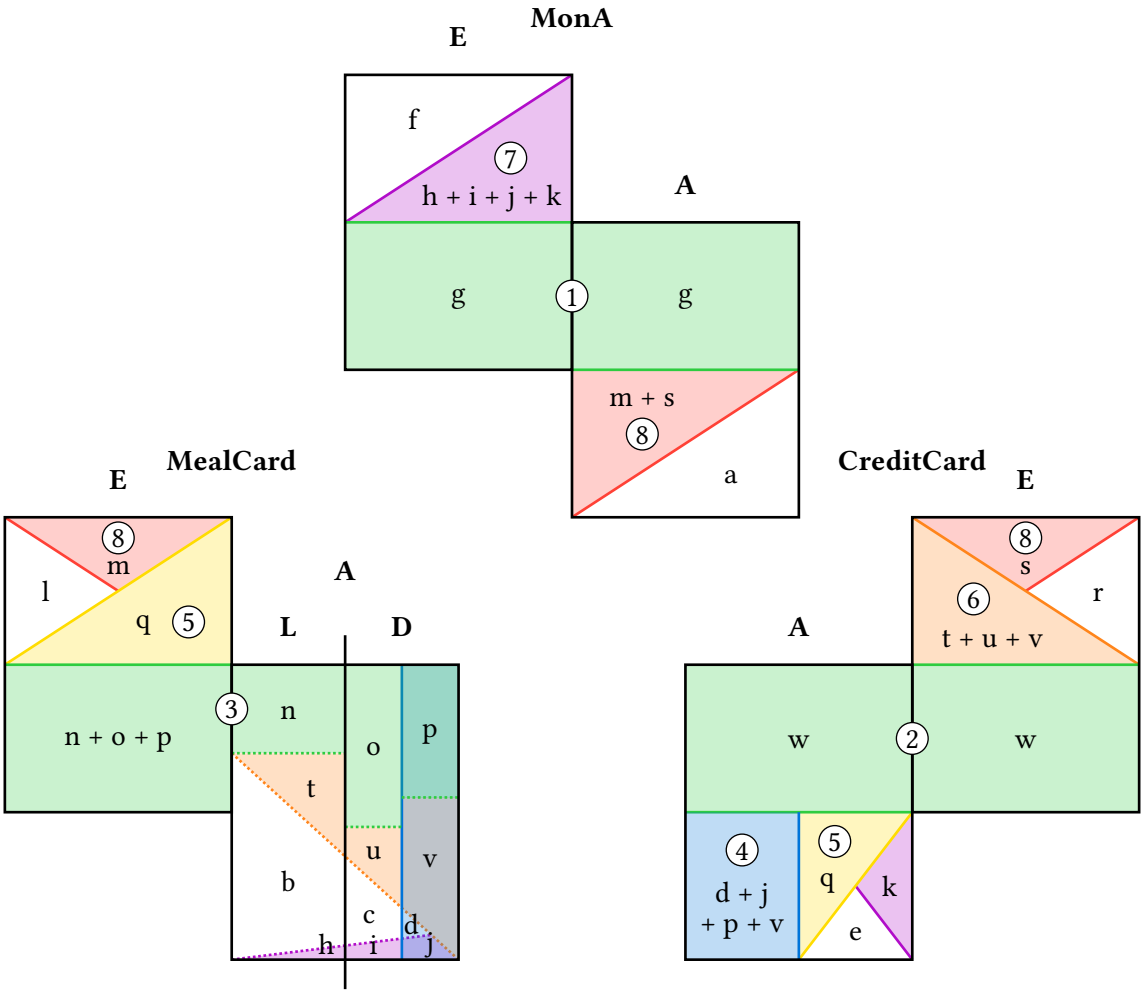
Ici il y a 4 lignes parce qu'une transaction peut être inattendue ou bien attendue en MonA, CB ou CBdèj'. Et il y a 6 colonnes parce qu'une transaction peut être non-reçue ou bien reçue en MonA, CB ou CBdèj', que si elle est reçue en CBdèj' elle peut être *différée* ou *directe*, et que si elle est directe elle peut être trouvée chez bpop en tant que CB ou non.

Ça nous donne tous les scénarios possibles pour une transaction dématérialisée. Par exemple, la situation `s` correspond à une transaction attendue en CB mais reçue en MonA (bilan nul pour la Cagette), et la situation `i` correspond à une transaction attendue en MonA, reçue en CBdèj' directe, mais pas trouvée chez bpop (bilan négatif pour la Cagette).

Le robot connait les nombres de transactions *marginaux*, c'est-à-dire les sommes des lignes et des colonnes, mais il ne peut pas en déduire les valeurs de toutes les cases. Le mieux qu'il peut faire, de nouveau, c'est de supposer que tout ce qui peut correspondre correspond et de maximiser les situations de bilan nul pour la Cagette. Cette fois-ci c'est beaucoup plus compliqué que dans le cas précédent, ça se passe dans le module `src/run/solve.rs` et ça fait appel à une librairie tierce d'optimisation linéaire.

Pour se faire une idée de comment ça fonctionne mais sans rentrer dans le détail, le plus intuitif c'est de jouer avec le solveur pour voir comment il réagit: https://cleb.io/prepabrinks/solver/.

Pour comprendre la logique utilisée, on peut la dessiner de cette façon:



On utilise la même représentation que précédemment, mais cette fois-ci il n'y a pas seulement la MonA, mais aussi les CB (l'attendu `E` est à droite) et les CBdèj' (le reçu `A` est divisé en deux flux *différé* `L` et *direct* `D`).

Les contraintes imposées au solveur implémentent la logique suivante:

- **(1) Correspondance MonA**: Le robot tâche de faire correspondre un maximum de transactions MonA ensemble en maximisant T_1, ça nous donne g.
- **(2) Correspondance CB**: Le robot tâche de faire correspondre un maximum de transactions CB ensemble en maximisant T_2, ça nous donne w.
- **(3) Correspondance CBdèj'**: Le robot tâche de faire correspondre un maximum de transactions CBdèj' ensemble en maximisant T_3, ça nous donne $n + o + p$, mais ça ne nous permet pas de savoir, parmis les transactions reçues correspondantes, lesquelles sont différées (n) ou bien directes-perdues (o) ou bien directes-reçues (p).
- **(4) Correspondance CBdèj' directes**: Le robot tâche de faire correspondre un maximum de transactions CBdèj' directes avec les CB inattendues en maximisant T_4. Ça fonctionne toujours sur le même principe, mais cette fois-ci le bloc "attendu" c'est `D` et le bloc "reçu" c'est l'ensemble des CB inattendues (`CCA - w`).

- **(5) Sauvetage CBdèj' - CB**: Les transactions manquantes CBdèj' peuvent se trouver chez bpop alors qu'elles ne sont pas encore apparues sur les sites des vendeurs. Ça arrive très souvent le vendredi par exemple parce que les sites des vendeurs sont lents à se mettre à jour. Le robot tâche alors faire correspondre un maximum de transactions CBdèj' manquantes avec ce qui reste des CB inattendues en maximisant T_5.
- **(6) Sauvetage CB - CBdèj'**: À l'inverse, des transactions CB manquantes peuvent se trouver sur les sites des vendeurs CBdèj' à la suite d'une erreur d'enregistrement en caisse. Le robot tâche alors de faire correspondre un maximum de transactions CB manquantes avec les CBdèj' inattendues en maximisant T_6. Ça ne nous dit pas s'il s'agit de transactions directes ou différées *etc.* (`t`, `u` ou `v`) mais c'est déjà un indice.
- **(7) Sauvetage MonA manquant**: De la même façon, suite à des erreurs de caisse, des paiements manquants en MonA peuvent se retrouver dans d'autres modalités. Le robot tâche alors de faire correspondre un maximum de transactions MonA manquantes avec ce qui reste des transactions CB ou CBdèj' inattendues en maximisant T_7.
- **(8) Sauvetage Mona inattendu**: Dans l'autre sens, le robot taĉhe de faire correspondre un maximum des transactions CB et CBdèj' manquantes restantes avec les transactions MonA inattendues, en maximisant T_8.

Plus formellement, ça se traduit par le système d'équations suivant, qu'il est important d'écrire correctement mais que l'on n'essaie pas de résoudre nous-mêmes :

(1) **MonA** $[E \times A]$

$$M_1 = f + h + i + j + k$$
$$T_1 = g$$
$$X_1 = a + m + s$$

(2) **CreditCard** $[E \times A]$

$$M_2 = r + s + t + u + v$$
$$T_2 = w$$
$$X_2 = d + j + p + v + e + k + q$$

(3) **MealCards** $[E \times (A = L + D)]$

$$M_3 = l + m + q \qquad T_3 = n + o + p$$
$$X_3 = b + c + d + h + i + j + t + u + v$$
$$L = b + h + n + t$$
$$D = c + d + i + j + o + p + u + v$$

(4) **Cross** $[D \times X_2]$

$$M_4 = c + i + o + u$$
$$T_4 = d + j + p + v$$
$$X_4 = e + k + q$$

(5) **Salvage** $[(\text{MealCards} = M_3) \times (X_4 = \text{CreditCard})]$

$$M_5 = l + m$$
$$T_5 = q$$
$$X_5 = e + k$$

(6) **Salvage** $[(\text{CreditCard} = M_2) \times (X_3 = \text{MealCards})]$

$$M_6 = r + s$$
$$T_6 = t + u + v$$
$$X_6 = b + c + d + h + i + j$$

(7) **Salvage** $[(\text{MonA} = M_1) \times (X_5 + X_6 = \text{MealCards} + \text{CreditCard})]$

$$M_7 = f$$
$$T_7 = h + i + j + k$$
$$X_7 = b + c + d + e$$

(8) **Salvage** $[(\text{MealCards} + \text{CreditCard} = M_5 + M_6) \times (X_1 = \text{MonA})]$

$$M_8 = l + r$$
$$T_8 = m + s$$
$$X_8 = a$$

On n'essaie pas de le résoudre parce que c'est compliqué et qu'on se doute bien qu'il n'existe pas toujours qu'une seule solution. Par exemple s'il y a 25€ manquants en MonA, 25€ inattendus en CBdèj' directe et 25€ inattendus en CBdèj' différée, on ne peut pas deviner comme ça lequel est qui même avec des super équations. Il faut faire une enquête.

En conséquence, parmi toutes les solutions qui respectent les contraintes données, le robot en choisit une arbitrairement selon le caprice du solveur, et se contente de reporter le résultat dans le fichier final en prenant soin d'interpeler le·la brinkseur·se si le bilan de la solution n'est pas équilibré pour la Cagette.

Une fois encore, le mieux pour comprendre sa décision est de s'approprier soi-même le comportement du solveur en le testant directement à l'adresse suivante : https://cleb.io/prepabrinks/solver/.

L'annotation

Une fois les composantes obtenues pour chaque modalité, les paquets de transactions uniformes établis entre les modalités, et le solveur invoqué pour chaque paquet uniforme, le robot dispose enfin de toutes les informations nécessaires pour parcourir les transactions une à une en produisant les commentaires associés si besoin et en collectant les anomalies. C'est une étape plutôt modulable qui se produit dans le fichier `src/run/annotate.rs`.

Le rendu

Une fois les annotations collectées, la totalité des évènements est de nouveau parcourue par le module `src/render.rs` qui se charge de construire le fichier de préparation résultant au format ouvert `.ods`. C'est un processus plutôt fastidieux à écrire parce que chaque cellule du tableau doit être dessinée indépendamment dans le code, et que le format du tableau de préparation attendu est plutôt fortement structuré avec les différentes sections *etc.* Mais la démarche générale est plutôt simple à comprendre : les colonnes sont d'abord préparées, vides, les formules introduites là où c'est nécessaire, puis le flux d'évènement + annotations + anomalies est lu dans l'ordre pour remplir le détail de chaque section.

À ce moment-là, le robot s'arrête et la prépa est prête pour révision par un·e humain·e brinkseur·ses :)