BEACHAIN

Is Code Law ?

L'archi-ultra-célèbre aphorisme de Lawrence Lessig « Code is law » (le code fait force de loi dans un monde numérisé : http://framablog.org/2010/05/22/code-is-law-lessig/) implique que l'utilisateur soit soumis par défaut - et en l'absence de corpus législatif adapté - à la loi imposée par l'algorithme. Celui-ci décrit un ensemble de règles, de processus et de fonctions dépendant de la vision stratégique de son codeur, et décide à situation donnée du résultat obtenu. L'homme n'aurait plus qu'à s'en accommoder à défaut d'y trouver un intérêt direct. « Code is law » signifie que l'interaction entre les hommes n'est plus médiatisée (au sens intermédiation) par des règles sociales consensuelles communes (rôle de la loi) mais par des décisions issues de calculs algorithmiques.

C'est à la fois vrai et faux. C'est vrai quand (et si...) dans l'exemple d'un smart-contact les conditions dans lesquelles il sera écrit entre les parties et accepté par les parties ne sont pas suffisamment précises ni suffisamment explicites. S'ensuivra alors nécessairement un certain nombre de choix laissés à l'algorithme et donc, par derrière, à l'appréciation du concepteur du code. Ce n'est donc pas « le code qui fait loi » mais le codeur lui-même.

Sous quelles conditions écrire alors un contrat de sorte que ce ne soit pas l'algorithme qui opère les choix une fois celui-ci signé, mais bien les parties elles-mêmes avant de le signer ? Posée autrement, la question revient à se demander comment *retirer la responsabilité de faire-loi* au codeur ?

La solution **beAchain**, basée sur des outils d'intelligence artificielle (IA), y répond en partie. Le rôle de l'IA est de déterminer lors de la rédaction d'un contrat les *zones à risques de mésinterprétations* pouvant créer un conflit potentiel. L'algorithme analyse le contexte transactionnel, vérifie la portée des conditions puis suggère des compléments si des imprécisions sont détectées.

Un exemple : je souhaite réserver un VTC pour me rendre Gare de l'Est à Paris afin de prendre un train à 18h02. Il est 17h10, le trafic est dense et le temps estimé, de là où je me trouve, est de 35 minutes. De là où je suis, le véhicule est annoncé pour les 3 minutes qui viennent. 35+3 = 38 minutes. L'arrivée Gare de l'Est étant prévue à 17h48 il me restera donc 14 minutes pour attraper mon train.

Lors de la rédaction du contrat quelques minutes plus tôt j'ai écrit :

[contexte] : *je voudrais une voiture pour la Gare de l'Est*

=> ULS : [JE] [MOVE] [mes coords GPS] [coord GPS Gare]\

Après validation entre objets de la chain (autres véhicules, autres machines connectées) j'ai le prix de la course et le temps d'arrivée du VTC ; le contrat est alors écrit dans un block et s'engage immédiatement. Quand le VTC sera à ma hauteur je monterai à bord puis, quand nous serons arrivés à destination - nos positions GPS respectives seront les mêmes que celles de la Gare - le versement du montant de la course sera instantanément réalisé de compte à compte après validation/consensus.

Sauf qu'il y a un problème : si pour un tas de raisons je rate mon train parce qu'on arrive à la gare après 18h, qui est responsable et que se passe-t-il ? Exemple de solution :

L'IA détecte qu'il y a déplacement (ULS [MOVE]) vers une destination. Elle demande alors des précisions :
[IA 1] *Si vous devez arriver à cette destination dans un certain délai indiquez les conditions en cas de réussite ou d'échec*

Je précise alors :

[case IF 1] : *si arrivée à la gare avant 17h50 =>*
=> code : *if (monGPS == GPSgare && time <= 17.4666)*
[case THEN 1] : *je règle le coût du transport*
=> code : *{ [JE] ([PAY] [VTC] [20]; }*
[case IF 2] : *si arrivée à la gare entre 17h50 et 18h*
=> code : *else if (monGPS == GPSgare && time > 17.4666 && time <= 18)*
[case THEN 2] : *je ne règle que la moitié du coût du transport*
=> code : *{ [JE] [PAY] [VTC] [10]; }*
[case IF 3] : *si arrivée à la gare après 18h*
=> code : *if (monGPS == GPSgare && time > 18)*
[case THEN 3] : *le véhicule paye le coût de modification de mon billet*
=> code : *{ [VTC] [PAY] [JE] [15]; }*

Le VTC ou son organisation (GOA, Groupe d'Objets associés) est libre d'accepter ou pas le marché et ses conditions contraignantes au prix qu'il veut. Une nouvelle proposition m'est faite : la voiture sera là dans 3 minutes mais la course sera de 25 euros (+5) que je suis à mon tour libre d'accepter ou pas.

Allons plus loin. L'IA peut demander ensuite si l'heure d'arrivée est impérative :
[IA 2] *Si vous devez arriver à cette destination dans un délai impératif indiquez les conditions en cas de réussite ou d'échec*

N'ayant plus d'autre train après celui de 18h04, le rater me coûterait une nuit d'hôtel. Je remplace la condition 3 par :

[case IF 3] : *si arrivée à la gare après 18h*
=> code : *if (monGPS == GPSgare && time > 18)*
[case THEN 3] : *le véhicule paye ma nuit d'hôtel*
=> code : *{ [VTC] [PAY] [JE] [80]; }*

Une nouvelle proposition m'est faite par le VTC ou son GOA : la course sera de 60 euros (x3) que je suis à mon tour libre d'accepter ou pas. L'engagement du VTC et son surcoût s'explique par l'amende statistiquement probable que sa propre IA aura calculée. Le risque de se faire arrêter est estimé à 1 sur 3 ; pour une amende de 120 euros le surcoût appliqué à la course est donc de 120/3. J'accepte ou je refuse la proposition.

Si j'accepte le contrat tel qu'il est écrit et codé, il est validé et écrit dans la blockchain. A partir de là tout est automatisé. Ce n'est bien sûr qu'un exemple fictif mais il montre comment des conditions peuvent s'empiler les unes aux autres jusqu'à décrire dans le détail chaque étape de sa réalisation.

Suite de l'exemple. S'il est passé 18h, l'intérêt du VTC est de s'arrêter net là où il est et de me faire sortir puisque m'emmener à destination lui coûterait 100 euros au final : les 80 de l'hôtel plus les 20 de manque à gagner sur la course. On adjoint une condition complémentaire pour éviter ce nouveau problème possible :

[case IF 4] : *si après 18h nous ne sommes toujours pas à la gare*
=> code : *if (monGPS != GPSgare && time > 18)*
[case THEN 4] : *le véhicule paye quand même la nuit d'hôtel + pénalité de 20 euros*
=> code : *{ [VTC] [PAY] [JE] [100]; }*

Il a donc intérêt à me déposer en retard plutôt que pas du tout.

Là le contrat est presque complet. Il manque encore une clause imposée par le VTC et suggérée par l'IA : si j'arrive à la gare avant 18h et que je paye la course, une fois monté dans le train il sera donc un peu plus de 18h. Le cas [IF 3] devrait alors automatiquement s'appliquer... à savoir que le VTC devrait me payer ma nuit d'hôtel. Donc :

[case IF 5] : *si la transaction a été effectuée quelle qu'elle soit*
=> code : *if (time > 18 && ([JE] [PAY] [VTC] || [VTC] [PAY] [JE]))*
[case THEN 5] : *on désactive le contrat*
=> code : *{ /end CONTRACT token; }*

On a donc tout intérêt dans un contrat complexe à conditions imbriquées en plusieurs couches à prendre le temps de bien le rédiger. Cela prend du temps mais beaucoup moins qu'un procès possible à venir.

Le code n'est pas la loi

La loi est l'acceptation par les deux parties de règles consenties que le code applique quelles qu'elles soient, donc telles qu'elles sont écrites. Il est de la responsabilité de chaque partie, assistée par l'IA, de s'assurer que toutes les conditions sont acceptables avant de signer le contrat les liant. Si une règle défavorable à l'une des deux parties est acceptée, sa responsabilité est engagée. *Nul ne peut contraindre une partie à signer un contrat qui ne lui convient pas*, sauf par des moyens qui dépassent l'application des smart-contracts d'une blockchain (par violence, par tromperie, par abus de faiblesse, etc.) et qui relèvent alors de la justice.

Vitalik Buterin, le créateur de Ethereum, a proposé l'an dernier la création d'un « tribunal blockchainé » (http://bitcoinist.net/vitalik-buterin-blockchain-court/). Sans rentrer dans les détails de ses propositions, il évoque entre autres idées celle de crowdsourcer l'arbitrage pour arriver à un genre de consensus : *arbitration in decentralized crowdsourcing and on-demand economy applications*. Un arbitrage ne peut pas être l'*opinion majoritaire* décidant qui a raison et qui a tort. Ce ne peut être qu'un jeu de règles préventives les plus précises possibles pour éviter tout recours à quelque arbitrage que ce soit. La généralisation de rédaction de contrats assistée par IA devrait avoir à éviter d'en passer par là.

Il existera toujours des paramètres qui échapperont aux capacités anticipatrices de l'IA ; pour les réduire, des technologies de *deep learning* seront progressivement implémentées dans **beAchain**. A mesure que des smart-contracts seront conclus entre parties ils viendront alimenter des séries de patterns modélisés pour entraîner le moteur d'IA à toujours plus et mieux anticiper. L'objectif final, en assistant la rédaction de contrats entre parties, est de parvenir à supprimer autant que faire se peut les risques de conflits entre elles. C'est la seule méthode pour démocratiser et faire adopter massivement les smart-contracts comme outils de médiation et de production pratiques et fiables.