The core exchange mechanism solves a fundamental problem of service exchange between a seller and a buyer, using the help of deposit repayment schemes and smart-contracts.

Formal Definition

There is a buyer, denoted by B

, and a seller, denoted by S

. The seller owns the item, denoted by I

. The item can be a digital file, a physical good, or a physical service. We design a mechanism, denoted by M

. The variables that are not known to the mechanism M

are:

- seller's valuation of the item, denoted by v_S

.

- buyer's valuation of the item, denoted by v_B

.

It must hold that $v_S<v_B$

, as otherwise, the exchange does not take place, since there is no positive value created.

The variables that are part of the mechanism are:

- price the buyer is willing to pay for the item, denoted by p

.

- deposit the seller is putting to the mechanism M

, denoted by D_S

.

- deposit the buyer is putting to the mechanism M

, denoted by D_B

.

It must hold that $v_S<p<v_B$

. Otherwise, the exchange does not take place, as it is not beneficial for both parties.

The core mechanism does not solve the problem of finding the price p

. This is a difficult problem. The seller wants to get the highest feasible price v_B

, while the buyer wants to pay the lowest feasible price v_S

. The problem is known as the double auction and extensively studied in the economic theory literature, Blumrosen and Dobzinski (2016). The mechanism leaves price determination to the seller and the buyer.

Two more variables are relevant to design the mechanism M

:

- cost of the seller to keep the item for the buyer, denoted by c_S

.

- cost of the buyer to pick-up the item from the seller, denoted by c_B

.

These parameters can be part of the design and provided by the players to the smart contract, as an input. At this iteration, both costs are assumed to be a certain fraction of the price, to keep it simple.

Both the buyer and the seller may offer the contract. The buyer may announce what item she wants, for how much and the seller agrees on it. However, depending on the application, only the seller might be able to offer it.

The current iteration of the core mechanism offers 3

different bits of information for the smart contract. Each bit of information corresponds to a move made by a player. That is, these information bits define final states, and the smart contract bases transfers to both participants on the final states.

- Redeem(R): move made by the buyer.
- Complaint(C): move made by the buyer.
- Cancel or Fault(CoF): move made by the seller.

All three messages have an explicit and intuitive explanation. Redeem is used to denote if the item/service was received or not. Complaint denotes if the service was high quality or not. The third move, CoF is probably the most involved one, and maybe even counter-intuitive, as the seller does not admit to fault even if it short-term beneficial for her. It is a reasonable assumption as sellers tend not to admit complaints when they are not at fault even if it is economically better because:

- they resent the unfairness.
- this would reward bad behavior, and therefore, create an incentive to repeat it.

The second holds because most of the sellers play a repeated game. If on the other hand it is a single-shot game, and the seller does not play it repeatedly, the assumption does not make sense. For such situations, other assumption that makes the mechanism work is required. In that case, we build a slightly different game, with a different order of moves and moves' logic. This is an example that the current mechanism does not deliver a ready solution in all situations. However, it can be adapted and fine-tuned depending on the application. At this point, the functioning of Cancel or Fault can be seen as totally rational. That is, we include it in the utility calculation, and solve the sequential game with correct, rational logic, taking into account this assumption.

Therefore, in this setting, there are 2^3=8

final states the contract can observe. Example state is denoted by s

. Mechanism M

has to specify 8\times 2=16

transfers, 2

transfers for each state s

. One transfer to the seller, denoted by t^{S}_s

, and one to the buyer, denoted by t^{B}_s

. T

denotes the set of transfers.

Note that the total number of real

states is more than 8

since there are hidden actions of the players. In this case, the seller might provide high or low-quality service, therefore, there are in total 16

states. For simplicity of exposition, we assume that the seller always provides an item or service. At the moment, the mechanism does not consider timing, or, it assumes that only one particular order is taking place. In some cases, this may not be a suitable approach. It might be better to have arbitrary sequences of messages, and for different orders, we may have different transfers, to optimize the performance of the mechanism. However, the analysis of the protocol gets more complicated and the implementation more involved.

Linear Programming Formulation

We specify conditions leading to a desired subgame-perfect equilibrium solution. The main idea is to have a solution that corresponds to the right play. That is, subgame-perfect equilibrium moves – optimal rational moves – are the same as honest moves.

Each decision at each internal

node of the decision tree gives a linear constraint on the variable set T

. These are so-called incentive compatibility constraints.

There are also budget-balance constraints

for each final state, namely that $t^s_S+t^s_B\leq D_S+D_B+p$

. Verbally it means that the mechanism can not add any money into the contract, but it can burn money.

Feasible linear programs normally have more than 1

solution. To pin down the solution set to one, we need to come up with the objective function. The first candidate is to minimize the sum of deposits. The friction coming from putting deposits has to be acknowledged. The obvious cost of putting high deposits for any (especially long) time period is opportunity cost. Depositing involves an implicit cost for the participants in the smart contract: This can be the opportunity cost of not using the deposit while the contract is executed, borrowing costs of the agent, risk of loss of the deposit if the consensus protocol fails, etc. In general, mechanisms that use punishment through huge negative transfers in case of "misbehavior" of agents are impractical. Even though the punishments are only executed off the equilibrium path, huge deposits have to be made to make the threat of punishment credible. This would make such mechanisms very costly to implement as a smart contract. Therefore, minimizing the sum of deposits can be an objective function. Opportunity cost in smart contracts and subgame perfect equilibria is a topic of Mamageishvili and Schlegel (2020). With instant exchange procedures, however, the opportunity cost is not of first-order importance.

An alternative objective function could be to maximize the money burnt (transferred to the protocol) off-equilibrium path. The transferred money could be used to analyze what went wrong in this particular game. This incentive for network nodes can be useful in the early stages of protocol functioning.

On figure 1 the green lines denote subgame-perfect equilibrium moves. The red lines denote off-equilibrium moves. Note that the first move is not observed by the mechanism. That is, the states in the left and right subtrees are the same, from the perspective of the smart contract.

In the first move of the seller, H corresponds to sending a high-quality item, while L corresponds to sending a low-quality item. In the second move, the buyer plays either redeem (R) or no redeem (refund or expire - RfE). In the third move, the buyer complains (C) or does not complain (NC). Finally, the seller acknowledges a fault (CF) or does not acknowledge the fault (NC).

Suppose the seller is deciding the root vertex S. Then the example incentive compatibility constraint at the vertex S

is the following:

$t_{S_8}^S + u^S_{H, R, NC,NC} \geq t_{S_1}^S + u^S_{L, RfE,C,CF}$,

where u^X_{p}

denotes the utility of player $X\in \{S,B\}$

when the mechanism takes path p

from the root to (any) leaf.

We consider quasi-linear utilities. That is, utility of the path for a player $X\in \{S,B\}$

in the decision tree is the sum of utilities of each move. Formally, $u_p^{X}=u_{e_1}+\cdots+u_{e_n}$

, where p

is a path consisting of edges $p=\{e_1,\cdots, e_n\}$

.

There are 15

internal nodes of the decision tree. Each node generates one incentive compatibility constraint of the linear program. Each final state adds 1

constraint. There are 2

additional {\it individual rationality} constraints, 1

for each player. In the subgame-perfect equilibrium solution, the expected payoff of the player should be non-negative. In other words, any player may opt-out for not participating in the contract. In total, there are $15+8+2=25$

constraints and 8

(transfer) variables. Optimizing target function given these constraints determines the values of transfers, and consequently, smart contract.

Figure 1:

ImgBB

decision-tree

Image decision-tree hosted in ImgBB

Discussion

Subgame-perfect implementation can be used with machine-to-machine commerce since it avoids the behavioral play of humans. Rational people program machines, therefore, there is no difference between these two. Machines are, on the other hand, better than irrational–behavioral players, or even rational players. Examples include but are not limited to that they do not press the wrong button by accident, they do not make decisions based on a bad mood (e.g, to punish other machines), they do not form opinions about other people's skills depending on how they look (this phenomenon is extensively studied in the experimental economic literature), etc. Programs just follow the rules rational people defined for them. Core exchange mechanism could be useful with machine-to-machine scenarios as long as machines can do proper verification of the service–item, that the seller provides. Examples include digital goods (files, codes, passwords), the example of a similar solution is Janin et al. (2020). With physical goods commerce, mechanism can still be used in the machine-to-machine approach. The only difference is that some steps (those that machines can not perform) will be performed by humans. Integrating machine and human is not the focus of the protocol. Another advantage of machine-to-machine use of the mechanism is that moves do not need to be intuitive and have a logical explanation. All the relevant information is final transfers and that the subgame-perfect equilibrium corresponds to the correct play. To design transfers in any game, we apply the same linear programming approach as in the previous section. We need to design how the subgame-perfect equilibrium solution looks, make appropriate assumptions on the utilities for both players coming from each move, and solve the linear program to find transfers in the final states leading to this subgame-perfect equilibrium solution.

Related Literature

Theoretical importance and applicability of subgame-perfect equilibria solutions were found much earlier. Moore and Repullo (1988) developed a general mechanism. The paper proposes subgame perfect implementation of most of the allocation problems given some mild conditions are satisfied. In other words, the authors prove that almost all social choice functions can be implemented as a subgame perfect equilibrium solution of some game. However, the game forms can be complicated, they have at most 3

stages.

A more recent experiment by Aghion et al. (2018) suggests that participants fail to play these games correctly and that often subgame-perfect equilibrium solution is not implemented. The authors argue that the problem is not in the inability of the participants to use backward induction for calculating the right solution, but in their beliefs about counterparts' abilities. 30\%

of the buyers who get a high-quality good lie, while 10\%

of the buyers lie all the time. In the context of the exchange mechanism, it would be the seller forming pessimistic beliefs about the buyer.

Two additional recent experimental papers, Fehr et al. (2020) and Chen et al. (2020) test theoretical predictions of how players should play if they were rational and believed that other players were also rational. The first one discusses the role of risk-aversion in sequential games. Players act irrationally if the mechanism introduces too high fines for misbehavior. The second one claims to have a simpler (2-stage) and more efficient (in the sense that players deviate from subgame-perfect play very rarely) mechanism.

These experimental papers explain why games with subgame-perfect equilibrium solutions are not implemented in real life, even though they are efficient and well understood in the theoretical economic literature. We could argue that one additional reason for this is the absence of low-cost intermediaries, which can be replaced by smart contracts and decentralized systems.

Failure of the Mechanism

In the following, we will discuss the experiment, where players fail to find subgame-perfect equilibrium, from Aghion et al. (2018). In the example there is a seller, who values the item/service as 0

, no matter the quality, a buyer who values item/service as 70

if it is high quality, and 20

if it is low quality. There are 4

stages, 3

observed by the mechanism. This example is an instantiation of the general mechanism from Moore and Repullo (1988).

1. Buyer sends either "high" or "low" message to the smart-contract. If the message is "high" and the seller does not "challenge" announcement, then the buyer pays the seller a price equal to 35

(which is the half of the high-quality item price), and the game ends.

1. If the buyer announces "low" and the seller does not "challenge" the buyer's announcement, then the buyer pays a price equal to 10

and the game ends.

1. If seller challenges buyer's announcement then:

3.1. Buyer pays a fine of F = 25

to T

(a third party, in our case the protocol).

3.2. Buyer is made a counter-offer for the good for 75

if his announcement was "high" and a price of 25

if his announcement was "low."

3.3. If the buyer accepts the counter-offer then the seller receives the fine F = 25

from T

(and also the counter-offer price from the buyer) and the game ends.

3.4. If the buyer rejects the counter-offer then the seller pays F = 25

to T

and the game ends.

Figure 2:

ImgBB

example

Image example hosted in ImgBB

Even though the game is quite intuitive and simple to solve, using backward induction, in around 30\%

of the cases, buyers send a wrong message when they receive high-quality service from the seller. However, the experiment is conducted in the lab and the participants do not know what is the right play in advance. Therefore, they first need to find the solution themselves and later form beliefs about their counterparts. The mechanism fails primarily because of the first moving player's beliefs about the next mover's abilities and the worst-case (utility) maximization. In other words, the first mover takes the safest move, not the optimal. In the core exchange mechanism, on the other hand, common knowledge is guaranteed by informing both parties joining the smart contract by default. Smart contracts can be seen as a way of generating common knowledge. Informally, common knowledge of a proposition P

means: every player knows P

, every player knows that every other player knows P

, every player knows that every other player knows that every player knows P

, and so on and so forth, infinitely. In our setting, P = \text{{all players are rational}}

.

This mechanism can be easily generalized to general values of h

and l

, where h

denotes the value of the high-quality item/service and l

denotes the value of the low-quality item/service. A similar generic example was developed by Gans (2019), with the difference that the seller has different costs of sending/providing different quality item/service.

Future Development

Experimenting exchange mechanisms with different parameters in different environments, assessing their performance, and finding the best contracts depending on the application is at the center of the core mechanism development. Creating a suitable environment for common knowledge in the game is the ultimate goal to achieve. Such an environment would allow the efficient deployment of not only exchange mechanisms but also more sophisticated economic mechanisms, which on its own would save intermediary fees and a negative impact powerful intermediaries or centralized authorities have on the general ecosystem. Natural live experiments using the core exchange mechanism should contribute to our knowledge in experimental economics as well. We have so far analyzed a game where only the seller makes a hidden move. In some applications, the buyer may also make a move, before they start reporting to the smart contract. This adds additional complexity to the analysis since the order of moves is not observed by the mechanism, therefore, the analysis of both possible sequences is required. We also need to include mechanism/smart contract fee calculation in the design and analysis.

Our goal is to design a game where rational play corresponds to honest play. This does not rule out malicious attackers. However, it guarantees that malicious attackers are punished for their harm to honest players, and therefore, a malicious attack is expensive.

References

Aghion, P., Fehr, E., Holden, R., and Wilkening, T. (2018). The role of bounded rationality and imperfect information in subgame perfect implementation–an empirical investigation. Journal of European Economic Association, 16(1):232–274.

Blumrosen, L. and Dobzinski, S. (2016). (almost) efficient mechanisms for bilateral trading. CoRR, abs/1604.04876.

Chen, Y.-C., Holden, R., Kunimoto, T., Sun, Y., and Wilkening, T. (2020). Getting dynamic implementation to work. Working Paper.

Fehr, E., Powell, M., and Wilkening, T. (2020). Behavioral constraints on the design of subgame-perfect implementation mechanisms. Forthcoming at American Economic Review.

Gans, J. S. (2019). The fine print in smart contracts. Working paper.

Janin, S., Qin, K., Mamageishvili, A., and Gervais, A. (2020). Filebounty: Fair data exchange. In IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2020, Genoa, Italy, September 7-11, 2020, pages 357–366. IEEE.

Mamageishvili, A. and Schlegel, J. C. (2020). Optimal smart contracts with costly verification. In IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020, Toronto, ON, Canada, May 2-6, 2020, pages 1–8. IEEE.

Moore, J. and Repullo, R. (1988). Subgame perfect implementation. 56(5):1191–1220. Econometrica,