# Structured REA contracts

Kasper Østerbye
IT University of Copenhagen

Non-formalized contracts can only be controlled, manipulated, and reasoned about by humans. Systems as well as humans can manage formalized contracts. This position paper presents a small extension of the REA ontology that formalizes contracts. The extension includes two new REA meta-types, and concretizes the existing Contract type. The proposed contract extension will allow REA based systems to monitor the state of REA contracts, and forms the basis for automating contract management and planning.

## 1 Introduction

In the REA ontology, the contract is a reification of the reciprocal relationship between commitments. A commitment is a promise or expectation of a future event. However, sometimes contracts involve choices to be made, e.g. pay $100 before may 1st, or accept a 10% penalty (in addition to the $100).

In [Peyton Jones, Eber, Seward, 2000] and [Peyton Jones, Eber, 2003] a contract specification language for financial contracts are proposed. We have adapted the language for REA.

The adaptation has introduced two new types into REA, Instantiation and Resolution. Intuitively, an instantiation is a generalization of a commitment. Instantiations are either commitments in traditional REA sense, they can be choices to be made, or they just serve a role in the internal structures of the solution. A Resolution is either an explicit entity that captures which specific choice was made, or it is just a structural entity.

An important aspect of Instantiation and Resolution is trace what options were available, and which options were picked.

## 2 Contract language

A contract is a static description that does not obligate anybody. This is opposed to an instantiated contract, which obligate the involved parties.

An example of a contract is shown in Figure 1.

```
(root KasperØsterbye PetersPaddles
  (exchange
      (transfer Paddle PetersPaddles KasperØsterbye)
      (anytime [now..August 2nd] KasperØsterbye
          (pick KasperØsterbye
              (anytime [now.. May 1st]KasperØsterbye
                  (transfer $300 KasperØsterbye PetersPaddles))
              (anytime [May 2nd  .. August 1st] KasperØsterbye
                  (transfer $330 KasperØsterbye PetersPaddles))
              (aytime August 2nd PetersPaddles
                  (apply collection $450
                      CaponeCollectors KasperØsterbye PetersPaddles)))
```

Figure 1 Contract for a paddle

The contract is a contract between me and PetersPaddels for a new paddle. The contract is an exchange between a paddle and some amount of money, which depends on when I pay. When the contract is instantiated, I get a paddle immediately, and sometime before august 2nd I have to choose payment. Before May 1st I have the option to pay $300, or to pay later.

No matter which option I choose, the payment chosen become the dual of the paddle.

The structure of the contract is so that at a given point in time, there are some parts of the contract which has been *instantiated*, while others still represent open options. In instantiation can be either pending, fulfilled, or expired.

The contract language has the following constructs.

(null)

The empty contract, nobody has promised anybody to do anything. The empty contract is always fulfilled.

(expired)

The expired contract can never be fulfilled.

(transfer ResourceDescription Provider Receiver)

The provider must transfer the Resources in ResourceDescription to the receiver. The transfer must take place immediately – I will return to this requirement later.

(pick Agent contract1 contract2 … contractN)

```
StandardDelayedPayment(Resource, Price, Customer, Deadline1, Deadline2) is
  (root Customer PetersPaddles
      (exchange
          (transfer Resource PetersPaddles Customer)
          (anytime [now..Deadline2+1] Customer
              (pick Customer
                  (anytime [now.. Deadline1] Customer
                      (transfer Price Customer PetersPaddles))
                  (anytime [Deadline1+1 .. Deadline2] Customer
                      (transfer Price+10% Customer PetersPaddles))
                  (anytime [Deadline2+1..Deadline2+1] PetersPaddles
                      (apply collection Price+50%
                      CaponeCollectors Customer PetersPaddles))))
```

Figure 2 Standard contract template.

The agent must immediately pick one of the subcontracts that are not expired. The chosen contract is instantiated.

(and contract1 contract2 …contractN)

All non-expired subcontracts are immediately instantiated.

(exchange contract1 contract2)

All the transfers that are the result of instantiating contract1 are reciprocal to those that are the result of contract2. Both contracts are instantiated immediately.

(anytime TimePeriod Agent contract)

The contract must be instantiated anytime in the time period; the given agent chooses the exact time.

(until Time contract)

The contract is immediately instantiated, but expires at the given Time.

(apply contractName arguments)

The contract template contractName is expanded with the given arguments. Contract templates are type images over contracts.

(root Agent1 Agent2 contract)

Root is the top of the contract, in which we declare who are involved. Contract is instantiated immediately.

As in the original papers by Peyton-Jones and Eber, the goal has been simplicity rather than appropriate level of abstraction. As an example, there is nothing in the above language that directly corresponds to a REA commitment. A REA commitment specifies a resource and a deadline. In our contract language this could be specified as a contract template:

```
REACommitment(ResourceDescription, Deadline,
Provider, Receiver)  is
  (anytime [now..deadline] Provider
      (transfer ResourceDescription Provider
Receiver))
```

## 3 Contract language applied

Assume that PetersPaddle's often sell paddle gear on contracts like in Figure 1. Then Peter will have a contract template of the form shown in Figure 2.

The expansion of this contract template with arguments Paddle, $300, KasperØsterbye, May 1st, August 1st become the contract from Figure 1.

When the root contract is instantiated, a paddle transfer commitment with immediate deadline is created, and an anytime choice commitment is also immediately created. We assume that the paddle transfer is immediately fulfilled by a sales event. The anytime choice can be instantiated anytime before the final deadline at the choice of the customer. When the customer chooses to instantiate the anytime choice, the customer must immediately pick which of the three options to pick. However, the customer cannot choose an expired contract.

By now, it ought to be clear that neither Peter nor the customer should work with these contracts directly. Some tool support is necessary. The primary tools include a contract translator, a choice assistant, and a deadline manager.

The choice assistant is should be able to calculate a future price for the contract depending on how the choices are made in the contract.

The contract translator is a tool that should be able to translate a contract into natural language. Such a tool is important to see if the contract is really what we want, and for customers to understand what the contract is all about.

The deadline manager should warn about upcoming choices, and their consequences (using the above two tools). The contract language has separated deadline and actions, so that one is awaiting a deadline, even an open-ended deadline (anytime between two dates are fine), but often there are implications further into the contract that has implications. For instance, the outermost anytime choice between now and deadline2 is seemingly a choice between three different things, but if one waits until the final day in the deadline, there is really no choice left.

## 4 Outline of operational semantics

It is important to distinguish between a contract and a contract instantiation. For each contract kind, there exist a corresponding instantiation. A contract in-

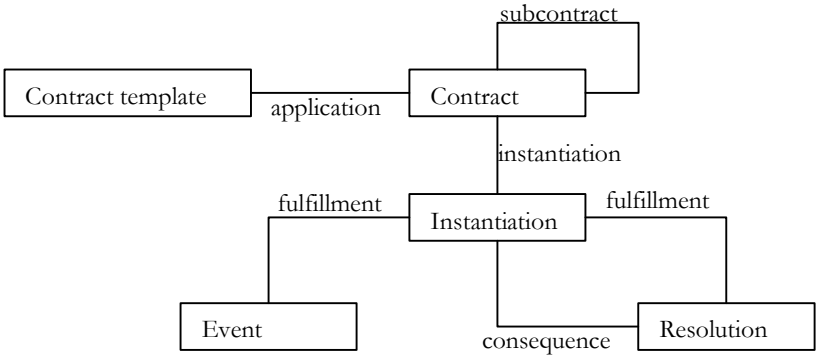


Figure 3 Extended Meta model

stantiation can be in one of three states, fulfilled, pending, or expired. See Figure 3.

The instantiation of a transfer contract will be viewed as a substitute for the REA commitment, and transfer instantiations can therefore be related to a REA event.

Resolution is a reification of a relationship between an instantiation, e.g. a pick instantiation, and the instantiations resulting from choosing a specific subcontract, or between an anytime instantiation, and the instantiation resulting from instantiating the contract at some point in time. The fulfilment relation specifies which resolution fulfilled the instantiation, and the consequence which new instantiations were created as a result. E.g. selecting a given pick resolves that pick, and has as consequence the instantiation of the selected pick.

An instantiation that has not been fulfilled is pending. If an instantiation is pending beyond its deadline, it is said to be expired, and it cannot become fulfilled. The null contract is instantiated into a null instantiation. The expired contract is instantiated into an expired instantiation.

The instantiation and resolution entities evolve over time into a tree structure, with the instantiation corresponding to the root contract at the top.

In this short paper, I shall not present the formalized semantics of instantiation, but try in the following to give a gist of the structure.

The semantics are about instantiation and resolution.

The instantiation semantics are rather simple, a contract of type root is instantiated into an instantiation of type root-instantiation, and a pick contract is instantiated into a pick-instantiation, the pick-instantiation contains a relationship to the possible contracts the choosing agent can pick from.

The interesting aspect is the resolution. A root-instantiation is resolved *immediately* and has as consequence that the contract argument is instantiated. A pick-contract must be resolved *immediately* by the agent, and is resolved by selecting one of the contracts from the pick instantiation.

The consequence is that the picked contract is instantiated. The and-contract is resolved *immediately* as well, and the consequence is that all sub-contracts are instantiated. The anytime instantiation is resolved anytime the agent decides it. The consequence is that the sub-contract is instantiated.

## 5 Conclusion

The proposed extension of REA does not change any of the existing concepts of REA, but gives a precise and expressive semantics to the REA contract. It allows the specification of contract templates, which can be expanded into contracts, which in turn can be signed and executed by the involved partners.

The examples in the present paper use fixed price, but often the price depends on the concrete usage of resources, and it is necessary to aggregate values from the operational part in REA into the contract language. For instance, a car repair has a cost which depends on the hours spend on the repair, and this must be included in the contract as well. This has been done in the semantic model, but is too elaborate to include in this paper.

## References

[Peyton Jones, Eber, Seward, 2000] S. Peyton Jones, J-M Eber, J. Seward. *Composing Contracts – an adventure in financial engineering.* International Conference on Functional Programming, Montreal, Sept 2000.

[Peyton Jones, Eber, 2003] S. Peyton Jones, J-M Eber. *How to write a financial contract.* In The Fun of Programming, edited by Gibbons and de Moor, Palgrave Macmillan 2003.