Retractable Contracts

Ivan Lanese
Computer Science Department
University of Bologna/INRIA
Italy

Joint work with Franco Barbanera,
Mariangiola Dezani-Ciancaglini
and Ugo de'Liguoro
Map of the talk

- Why retractable contracts?
- What is a retractable contract?
- Results
- Conclusion
Map of the talk

- Why retractable contracts?
- What is a retractable contract?
- Results
- Conclusion
Contracts

- A contract is the abstract description of the behavior of either a client or a server.
- A client complies with a server if all her requirements are fulfilled:
  - by reaching a distinguished satisfaction state or
  - by running an infinite interaction without ever getting stuck.
- A client that does not comply with its server may get stuck.
- Compliance is statically decidable.
Undoing things considered harmful

- Undo operations are useful and widespread
  - Undo command in your favorite editor
  - Restore a past backup
  - Back button in your favorite browser

- In interacting systems (unilateral) undo may lead to unpredictable or undesired results
  - What happens if you press the back button when reserving a flight?
  - You don’t want a client to undo her payment after a purchase

- Undo activities must be disciplined

- Retractable contracts are a way to discipline activities including undo operations
Retractable contracts: approach

- Getting stuck may depend on wrong choices taken during the interaction
- Going back to past choices and trying different paths may solve the problem
- This will “facilitate” compliance
- In this work we explore a notion of contracts where past decisions are stored and can be undone
Map of the talk

- Why retractable contracts?
- What is a retractable contract?
- Results
- Conclusion
Retractable contracts: syntax

\[ \sigma ::= 1 \quad \text{success} \]
\[ \bigoplus_{i \in I} a_i . \sigma \quad \text{internal output choice} \]
\[ \Sigma_{i \in I} a_i . \sigma \quad \text{external input choice} \]

\[ X \quad \text{variable} \]
\[ \text{rec } X.\sigma \quad \text{recursion} \]
\[ \Sigma_{i \in I} \bar{a}_i . \sigma \quad \text{retractable output choice} \]
\[ \bigoplus_{i \in I} a_i . \sigma \quad \text{internal input choice} \]

Standard contracts
Retractable contracts: main idea

- The peculiar operator is retractable output choice: \[ \sum_{i \in I} a_i \sigma_i \]
- It behaves as follows:
  - it performs an output, but other options are stored
  - if the computation gets stuck, undo is performed and another option is tried
Retractable contracts: history information

- To give semantics to contracts we need history information
- We add $\diamond$ (empty contract) to contracts $\sigma$
- Histories are stacks of contracts $h ::= [] \mid h:\sigma$
- Contracts with history: $h \prec \sigma$
Motivating problem

- A buyer wants to buy either a bag or a belt
- She will decide whether to pay by card or cash after knowing the price
- \[ \text{Buyer} = \text{bag.price.}(\text{card} \oplus \text{cash}) \oplus \text{belt.price.}(\text{card} \oplus \text{cash}) \]
- The seller accepts cards only for bags, not for belts
- \[ \text{Seller} = \text{bag.price.}(\text{card} + \text{cash}) + \text{belt.price.cash} \]
- Buyer and seller are not compliant
Reversibility to the rescue

- **Buyer** = 
  \[ \text{bag.price.(card } \oplus \text{ cash) } \oplus \text{ belt.price.(card } \oplus \text{ cash)} \]

- **Seller** = 
  \[ \text{bag.price.(card } + \text{ cash)} + \text{ belt.price.cash} \]

- They become compliant if we make the buyer choice between bag and belt retractable
  - Or the one between card and cash (for belt)

- The buyer is still able to pay a belt with card if interacting with a seller allowing this
Reversibility to the rescue

- **Buyer** = \(\text{bag.price.(card} \oplus \text{cash)} + \text{belt.price.(card} \oplus \text{cash)}\)

- **Seller** = \(\text{bag.price.(card} + \text{cash)} + \text{belt.price.cash}\)

- They become compliant if we make the buyer choice between bag and belt retractable
  - Or the one between card and cash (for belt)

- The buyer is still able to pay a belt with card if interacting with a seller allowing this
Sample computation

- **Buyer’ =**
  
  \[
  [] \xleftarrow{\text{bag.price.}(\text{card} \oplus \text{cash}) + \text{belt.price.}(\text{card} \oplus \text{cash})}
  \]

- **Seller =**
  
  \[
  [] \xleftarrow{\text{bag.price.}(\text{card} + \text{cash}) + \text{belt.price.}\text{cash}}
  \]
Sample computation

- **Buyer’ =**
  
  \[
  \emptyset \prec \text{bag.price.(card} \oplus \text{cash}) + \text{belt.price.(card} \oplus \text{cash})
  \]

  \[\text{bag.price.(card} \oplus \text{cash}) \prec \text{price.(card} \oplus \text{cash})\]

- **Seller =**
  
  \[
  \emptyset \prec \text{bag.price.(card} + \text{cash}) + \text{belt.price.cash}
  \]

  \[\text{bag.price.(card} + \text{cash}) \prec \text{price.cash}\]
Sample computation

- **Buyer’ =**
  
  \[
  \emptyset \prec \text{bag} \cdot \text{price} \cdot (\text{card} \oplus \text{cash}) + \text{belt} \cdot \text{price} \cdot (\text{card} \oplus \text{cash})
  \]

  \[
  \text{bag} \cdot \text{price} \cdot (\text{card} \oplus \text{cash}) \prec \text{price} \cdot (\text{card} \oplus \text{cash})
  \]

  \[
  \text{bag} \cdot \text{price} \cdot (\text{card} \oplus \text{cash}) : \circ \prec \text{card} \oplus \text{cash}
  \]

- **Seller =**

  \[
  \emptyset \prec \text{bag} \cdot \text{price} \cdot (\text{card} + \text{cash}) + \text{belt} \cdot \text{price} \cdot \text{cash}
  \]

  \[
  \text{bag} \cdot \text{price} \cdot (\text{card} + \text{cash}) \prec \text{price} \cdot \text{cash}
  \]

  \[
  \text{bag} \cdot \text{price} \cdot (\text{card} + \text{cash}) : \circ \prec \text{cash}
  \]
Sample computation

- **Buyer’ =**
  \[
  [] \prec \text{bag.
  price.}(\text{card} \oplus \text{cash}) + \text{belt.
  price.}(\text{card} \oplus \text{cash})
  \]
  \[
  \text{bag.
  price.}(\text{card} \oplus \text{cash}) \prec \text{price.}(\text{card} \oplus \text{cash})
  \]
  \[
  \text{bag.
  price.}(\text{card} \oplus \text{cash}) : \circ \prec \text{card} \oplus \text{cash}
  \]
  \[
  \text{bag.
  price.}(\text{card} \oplus \text{cash}) : \circ \prec \text{card}
  \]

- **Seller =**
  \[
  [] \prec \text{bag.
  price.}(\text{card} + \text{cash}) + \text{belt.
  price.}\text{cash}
  \]
  \[
  \text{bag.
  price.}(\text{card} + \text{cash}) \prec \text{price.}\text{cash}
  \]
  \[
  \text{bag.
  price.}(\text{card} + \text{cash}) : \circ \prec \text{cash}
  \]
Sample computation

- Buyer' = bag . price . (card ⊕ cash) : o < card

- Seller = bag . price . (card + cash) : o < cash
Sample computation

- **Buyer’ =**
  \[ \text{bag.price.}(\text{card} \oplus \text{cash}) : \text{o} \prec \text{card} \]
  - \[ \text{bag.price.}(\text{card} \oplus \text{cash}) \prec \text{o} \]

- **Seller =**
  \[ \text{bag.price.}(\text{card} + \text{cash}) : \text{o} \prec \text{cash} \]
  - \[ \text{bag.price.}(\text{card} + \text{cash}) \prec \text{o} \]
Sample computation

- **Buyer'** =
  \[
  \text{bag.price.}(\text{card} \oplus \text{cash}) : \circ \prec \text{card}
  \]
  \[
  \text{bag.price.}(\text{card} \oplus \text{cash}) \prec \circ
  \]
  \[
  [] \prec \text{bag.price.}(\text{card} \oplus \text{cash})
  \]

- **Seller** =
  \[
  \text{bag.price.}(\text{card} + \text{cash}) : \circ \prec \text{cash}
  \]
  \[
  \text{bag.price.}(\text{card} + \text{cash}) \prec \circ
  \]
  \[
  [] \prec \text{bag.price.}(\text{card} + \text{cash})
  \]
Sample computation

- **Buyer’ =**
  \[ [] \leftarrow \text{bag.price.}(\text{card} \oplus \text{cash}) \]

- **Seller =**
  \[ [] \leftarrow \text{bag.price.}(\text{card} + \text{cash}) \]
Sample computation

- **Buyer’ =**
  
  $$\langle \text{[]} \triangleleft \text{bag.price.(card } \oplus \text{ cash)} \rangle$$  

  - $$\circ \triangleleft \text{price.(card } \oplus \text{ cash)}$$

- **Seller =**
  
  $$\langle \text{[]} \triangleleft \text{bag.price.(card } + \text{ cash)} \rangle$$

  - $$\circ \triangleleft \text{price.(card } + \text{ cash)}$$
Sample computation

• **Buyer’ =**
  
  \[
  \emptyset \prec \text{bag.price.}(\text{card} \oplus \text{cash})
  \]
  
  ○ \prec \text{price.}(\text{card} \oplus \text{cash})
  
  ○ : ○ \prec \text{card} \oplus \text{cash}

• **Seller =**
  
  \[
  \emptyset \prec \text{bag.price.}(\text{card} + \text{cash})
  \]
  
  ○ \prec \text{price.}(\text{card} + \text{cash})
  
  ○ : ○ \prec \text{card} + \text{cash}
Sample computation

- **Buyer'** = ______________________
  ○ : ○ ↘ card ⊕ cash

- **Seller** =
  ○ : ○ ↘ card + cash
Sample computation

- **Buyer’ =**
  - ○ : ○ \(\prec\) card \(\oplus\) cash
  - ○ : ○ \(\prec\) card

- **Seller =**
  - ○ : ○ \(\prec\) card + cash
Sample computation

- **Buyer’ =**
  - ○ : ○ \(\prec\) card \(\oplus\) cash
- ○ : ○ \(\prec\) card
- ○ : ○ : ○ \(\prec\) 1

- **Seller =**
  - ○ : ○ \(\prec\) card + cash
- ○ : ○ : cash \(\prec\) 1
Map of the talk

- Why retractable contracts?
- What is a retractable contract?
- Results
- Conclusion
Compliance

- The compliance relation $h \prec \sigma \parallel k \prec \rho$ holds iff $h \prec \sigma \parallel k \prec \rho \rightarrow^* h' \prec \sigma' \parallel k' \prec \rho' \not\leftrightarrow$ implies $\sigma' = 1$
  - If the computation stops then the client is satisfied

- The compliance relation on contracts is obtained by executing them with an empty history
Compliance: results

- Compliance is decidable even for contracts with recursion
- The complexity is $O(n^5)$
  - Straightforward algorithm is exponential
- The algorithm extends in a non trivial way the one for subtyping of recursive arrow and product types from Pierce
Subcontract relation

- Subcontract relation for servers:
  \( \rho \preceq_s \rho' \) iff for each client \( \sigma \).
  \( \sigma \models \rho \) implies \( \sigma \models \rho' \)
  - \( \rho \) has more clients than \( \rho' \)

- Subcontract relation for clients is dual:
  \( \sigma \preceq_c \sigma' \) iff for each server \( \rho \).
  \( \sigma \models \rho \) implies \( \sigma' \models \rho \)

- The two subcontract relations are partial orders

- The dual \( \bar{\sigma} \) of a client contract \( \sigma \) is the minimum server
  compliant with \( \sigma \)
Subcontract relation: example
Duality has a simple syntactic characterization

<table>
<thead>
<tr>
<th></th>
<th>⇔</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>⇔</td>
<td>1</td>
</tr>
<tr>
<td>[ \Sigma_{i \in I} a_i \sigma_i ]</td>
<td>⇔</td>
<td>[ \oplus_{i \in I} a_i \sigma_i ]</td>
</tr>
<tr>
<td>[ \Sigma_{i \in I} \bar{a}_i \sigma_i ]</td>
<td>⇔</td>
<td>[ \oplus_{i \in I} a_i \sigma_i ]</td>
</tr>
<tr>
<td>[ X ]</td>
<td>⇔</td>
<td>[ X ]</td>
</tr>
<tr>
<td>[ \text{rec } X.\sigma ]</td>
<td>⇔</td>
<td>[ \text{rec } X.\sigma ]</td>
</tr>
</tbody>
</table>
Subcontract relation: results

- Subcontract relation for servers and for clients are related:
  \[ \rho \preceq_s \rho' \iff \bar{\rho}' \preceq_c \bar{\rho} \]

- Subcontract relation and compliance are related:
  \[ \rho \preceq_s \rho' \iff \bar{\rho} \parallel \| \rho' \]

- Also the subcontract relation can be decided in \( O(n^5) \)
Retractable contracts vs reversible computation

- Take retractable contracts without retraction
- Apply to it the technique to make a calculus reversible from Phillips and Ulidowski
- Retraction corresponds to a sequence of backward steps in the resulting reversible calculus
- Hence, retractable contracts are a form of reversible computation with internal/semantic control
- If you drop these forms of control then compliance becomes trivial
Map of the talk

- Why retractable contracts?
- What is a retractable contract?
- Results
- Conclusion
Summary

- We presented a model of contracts with retractable choice.
- Using retractable choice instead of normal choice ensures compliance with a larger set of partners.
- Retractable contracts have most of the good properties of contracts:
  - decidability of compliance and subcontract relation
  - efficient decidability algorithm
  - easy syntactic characterization of duality
Future work

- Explore the notion of retractable contracts in multiparty sessions
- How can we extract a contract from a reversible application?
- Are there other meaningful ways to exploit contracts/behavioural types to control reversibility?
End of talk

Thanks!

Questions?
Most related work

- Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ugo de'Liguoro: Compliance for reversible client/server interactions. BEAT 2014
- also considered contracts with rollback

<table>
<thead>
<tr>
<th>BEAT 2014</th>
<th>vs</th>
<th>PLACES 2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free rollback</td>
<td>vs</td>
<td>rollback only when stuck</td>
</tr>
<tr>
<td>Explicit checkpoint</td>
<td>vs</td>
<td>implicit checkpoint</td>
</tr>
<tr>
<td>One checkpoint</td>
<td>vs</td>
<td>stack of checkpoints</td>
</tr>
<tr>
<td>Compliance harder</td>
<td>vs</td>
<td>compliance easier</td>
</tr>
</tbody>
</table>