Retractable Contracts And Beyond

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?
- What is a speculative contract?
- What is beyond?
Map of the talk

- Why retractable contracts?
- What is a retractable contract?
- What is a speculative contract?
- What is beyond?
Undoing things considered harmful

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

- **In interactions (unilateral) undo may lead to unpredictable or undesired results**
  - What happens if you press the back button of the browser while reserving a flight?
  - You don’t want a client to be able to undo her payment after a purchase

- **Undo activities must be disciplined**
Contracts

- A (binary) **contract** is the abstract description of the behaviour of 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
Retractable contracts

• We start from binary contracts
• 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
• 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?
- What is a speculative contract?
- What is beyond?
Retractable contracts: syntax

- $\sigma, \rho ::= 1$  
  success
- $\sum_{i \in I} a_i \cdot \sigma_i$  
  external input choice
- $\Theta_{i \in I} \overline{a}_i \cdot \sigma_i$  
  internal output choice
- $\sum_{i \in I} \overline{a}_i \cdot \sigma_i$  
  retractable output choice
- $\Theta_{i \in I} a_i \cdot \sigma_i$  
  internal input choice
- $x$  
  variable
- $rec \ x \cdot \sigma$  
  recursion

- We add $\circ$ (no more alternatives) to contracts $\sigma$
- Histories are stacks of contracts $h ::= [\ ] | h : \sigma$
- Contracts with history: $h < \sigma$
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

\[
\text{Buyer} = \overline{\text{bag}} \cdot \text{price} \cdot (\overline{\text{card}} \oplus \overline{\text{cash}}) \oplus \overline{\text{belt}} \cdot \text{price} \cdot (\overline{\text{card}} \oplus \overline{\text{cash}})
\]

- The seller accepts cards only for bags, not for belts

\[
\text{Seller} = \text{bag} \cdot \overline{\text{price}} \cdot (\text{card} + \text{cash}) + \text{belt} \cdot \overline{\text{price}} \cdot \text{cash}
\]

- Buyer and seller are not compliant
- They become compliant if we make, e.g., the buyer choice between bag and belt retractable
- The buyer is still able to pay a belt with card if interacting with a seller allowing this
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

\[ Buyer = \overline{bag} \cdot price \cdot (\overline{card} \oplus \overline{cash}) + \overline{belt} \cdot price \cdot (\overline{card} \oplus \overline{cash}) \]

- The seller accepts cards only for bags, not for belts

\[ Seller = bag \cdot price \cdot (card + cash) + belt \cdot price \cdot cash \]

- Buyer and seller are not compliant
- They become compliant if we make, e.g., the buyer choice between bag and belt retractable
- The buyer is still able to pay a belt with card if interacting with a seller allowing this
Retractable contracts: semantics

- Contracts are executed as usual but...
- … branches in external choices which are not selected are stored in the history
- When the interaction is stuck, both client and server can pop from the history the last state
Sample computation

Buyer =
\[ [] < \overline{bag} \cdot \overline{price} \cdot (\overline{card} \oplus \overline{cash}) + \overline{belt} \cdot \overline{price} \cdot (\overline{card} \oplus \overline{cash}) \]

Seller =
\[ [] < \overline{bag} \cdot \overline{price} \cdot (\overline{card} + \overline{cash}) + \overline{belt} \cdot \overline{price} \cdot \overline{cash} \]
Sample computation

- **Buyer =**
  \[
  \{ \} < bag \cdot price \cdot (card \oplus cash) + belt \cdot price \cdot (card \oplus cash)
  \]
  \[
  \Rightarrow \{ \} : bag \cdot price \cdot (card \oplus cash) < price \cdot (card \oplus cash)
  \]

- **Seller =**
  \[
  \{ \} < bag \cdot price \cdot (card + cash) + belt \cdot price \cdot cash
  \]
  \[
  \Rightarrow \{ \} : bag \cdot price \cdot (card + cash) < price \cdot cash
  \]
Sample computation

• Buyer =

  \[
  [] < \text{bag} \cdot \text{price} (\text{card} \oplus \text{cash}) + \text{belt} \cdot \text{price} (\text{card} \oplus \text{cash})
  \]

  \rightarrow

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

  \rightarrow

  \[
  [] : \text{bag} \cdot \text{price} (\text{card} \oplus \text{cash}) : \odot < \text{card} \oplus \text{cash}
  \]

• Seller =

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

  \rightarrow

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

  \rightarrow

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

Buyer =

\[
[]: \text{bag}. \text{price}. (\text{card} \oplus \text{cash}) : \circ < \text{card} \oplus \text{cash}
\]

Seller =

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

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

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

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

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

  \[
  \rightarrow []: \overline{\text{bag}. \text{price}. (\overline{\text{card}} \oplus \overline{\text{cash}})} : \circ < \overline{\text{card}}
  \]

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

Interaction is stuck
Sample computation

• Buyer =
  \[ []: \overline{bag}. price . (\overline{card} \oplus \overline{cash}) : \circ < \overline{card} \]

• Seller =
  \[ []: \overline{bag}. price . (\overline{card} + \overline{cash}) : \circ < \overline{cash} \]
Sample computation

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

  \[\rightarrow \text{[ ]: } \text{bag}. \text{price}. (\text{card} \oplus \text{cash}) \ < \ \circ\]

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

  \[\rightarrow \text{[ ]: } \text{bag}. \text{price}. (\text{card} + \text{cash}) \ < \ \circ\]
Sample computation

**Buyer =**

\[
\begin{align*}
\text{[]: } & \text{\underline{bag}. price. (\underline{card} \oplus \underline{cash})}: \circ < \underline{\text{card}} \\
\rightarrow & \text{[]: } \text{\underline{bag}. price. (\underline{card} \oplus \underline{cash})} < \circ \\
\rightarrow & \text{[]} < \text{\underline{bag}. price. (\underline{card} \oplus \underline{cash})}
\end{align*}
\]

**Seller =**

\[
\begin{align*}
\text{[]: } & \text{\underline{bag}. price. (\underline{card} + \underline{cash})}: \circ < \underline{\text{cash}} \\
\rightarrow & \text{[]: } \text{\underline{bag}. price. (\underline{card} + \underline{cash})} < \circ \\
\rightarrow & \text{[]} < \text{\underline{bag}. price. (\underline{card} + \underline{cash})}
\end{align*}
\]
Sample computation

- Buyer =
  \[ [] < \overline{bag\. price} \cdot (\overline{card} \oplus \overline{cash}) \]

- Seller =
  \[ [] < \overline{bag\. price} \cdot (\overline{card} + \overline{cash}) \]
Sample computation

• Buyer =

  [ ] < bag . price . (card ⊕ cash)

  → [ ] : ∘ < price . (card ⊕ cash)

• Seller =

  [ ] < bag . price . (card + cash)

  → [ ] : ∘ < price . (card + cash)
Sample computation

• Buyer =
  
  \[ \emptyset < \text{bag}.\text{price}.(\text{card} \oplus \text{cash}) \]
  
  \[ \rightarrow \emptyset : \circ < \text{price}.(\text{card} \oplus \text{cash}) \]
  
  \[ \rightarrow \emptyset : \circ : \circ < \text{card} \oplus \text{cash} \]

• Seller =
  
  \[ \emptyset < \text{bag}.\text{price}.(\text{card} + \text{cash}) \]
  
  \[ \rightarrow \emptyset : \circ < \text{price}.(\text{card} + \text{cash}) \]
  
  \[ \rightarrow \emptyset : \circ : \circ < \text{card} + \text{cash} \]
Sample computation

- **Buyer =**
  \[ []: \circ: \circ \prec \overline{\text{card}} \oplus \overline{\text{cash}} \]

- **Seller =**
  \[ []: \circ: \circ \prec \text{card} + \text{cash} \]
Sample computation

- Buyer =
  [ ]:○:○ < card ⊕ cash
→ [ ]:○:○ < card

- Seller =
  [ ]:○:○ < card + cash
Sample computation

- **Buyer =**
  
  \[
  \text{[]} : \circ : \circ < \text{card} \oplus \text{cash}
  \]

  \[\rightarrow \text{[]} : \circ : \circ < \text{card} \]

  \[\rightarrow \text{[]} : \circ : \circ : \circ < 1\]

- **Seller =**
  
  \[
  \text{[]} : \circ : \circ < \text{card} + \text{cash}
  \]

  \[
  \text{[]} : \circ : \circ : \text{cash} < 1
  \]
Compliance

- We can use the standard notion of compliance
  - If the computation gets stuck, then the client is satisfied
- We can define a formal system to decide compliance
- The main novelty is that two external choices are compliant
  iff there exists a compatible branch
  \[ \Gamma, \alpha.\rho + \rho' \vdash \alpha.\sigma + \sigma' \Downarrow \rho \vdash \sigma \]

\[ \Gamma \Downarrow \alpha.\rho + \rho' \vdash \alpha.\sigma + \sigma' \]

- The formal system is correct, complete and terminating,
  hence it can be transformed into a procedure
- The procedure requires \( O(n^5) \)
  - More than for standard contracts
Duality and subcontract relation

- From the notion of compliance we can define a notion of subcontract
  - Replacing a contract with a subcontract preserves compliance
- The syntactic dual is also a semantic dual
  - The more general compliant contract
Map of the talk

- Why retractable contracts?
- What is a retractable contract?
- What is a speculative contract?
- What is beyond?
Speculative contracts

- We start again from binary contracts
- For efficiency reasons one may want to try different options concurrently
- As soon as one of them succeeds, the whole computation is successful
- We use the same syntax that we used for retractable contracts
  - Now external choice among outputs has the speculative behaviour above
  - External since the environment can slow down undesired paths selecting the one he wants to succeed
Speculative contracts: results

- The decision procedure for retractable contracts and for speculative contracts coincide
- As a consequence, all the results about compliance, duality and subcontract apply
Map of the talk

- Why retractable contracts?
- What is a retractable contract?
- What is a speculative contract?
- What is beyond?
Summary

- We presented a model of contracts with retractable choice
  - Compliance
  - Subcontract relation
  - Duality
- Simple and neat extension of the theory of binary contracts
- Using retractable choice instead of normal choice extends the set of compliant contracts
- The same theory captures also speculative execution
What is beyond?

- Explore the notion of retractable contracts in multiparty sessions
- Are there other meaningful ways to exploit contracts/behavioural types to control reversibility?
- Are there other useful computational patterns that can be tamed using contracts?
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

BEAT 2014    vs    PLACES 2015
- Free rollback    vs    rollback only when stuck
- Explicit checkpoint    vs    implicit checkpoint
- One checkpoint    vs    stack of checkpoints
- Compliance harder    vs    compliance easier