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

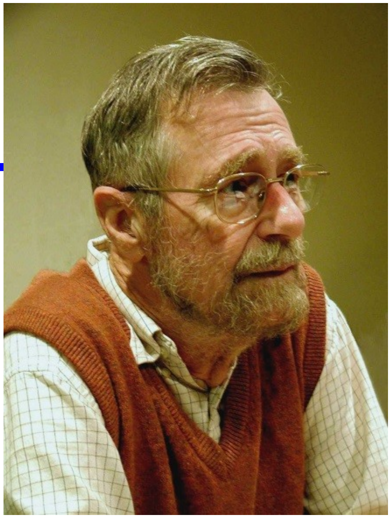
Map of the talk

- 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

$$
\begin{array}{lll}
\sigma ::= & \left.\begin{array}{ll}
1 & \text{success} \\
\bigoplus_{i\in I} \overline{a}_i.\sigma_i & \text{internal output choice} \\
\Sigma_{i\in I}\, a_i.\sigma_i & \text{external input choice} \\
X & \text{variable} \\
\mathrm{rec}\; X.\sigma & \text{recursion}
\end{array}\right\} & \begin{array}{l}\text{Standard} \\ \text{contracts}\end{array} \\
 & \begin{array}{ll}
\Sigma_{i\in I}\, \overline{a}_i.\sigma_i & \text{retractable output choice} \\
\bigoplus_{i\in I} a_i.\sigma_i & \text{internal input choice}
\end{array} &
\end{array}
$$

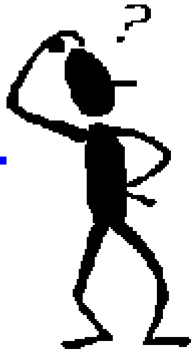
Retractable contracts: main idea

- The peculiar operator is retractable output choice: $\Sigma_{i\in I}\ \overline{a}_i.\sigma_i$
- 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 ○ (empty contract) to contracts σ
- 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
- Buyer =

  $\overline{\mathrm{bag}}.\mathrm{price}.(\overline{\mathrm{card}} \oplus \overline{\mathrm{cash}}) \oplus \overline{\mathrm{belt}}.\mathrm{price}.(\overline{\mathrm{card}} \oplus \overline{\mathrm{cash}})$
- The seller accepts cards only for bags, not for belts
- Seller =

  $\mathrm{bag}.\overline{\mathrm{price}}.(\mathrm{card} + \mathrm{cash}) + \mathrm{belt}.\overline{\mathrm{price}}.\mathrm{cash}$
- Buyer and seller are not compliant

Reversibility to the rescue

- Buyer =
  $\overline{\mathrm{bag}}.\mathrm{price}.(\overline{\mathrm{card}} \oplus \overline{\mathrm{cash}}) \oplus \overline{\mathrm{belt}}.\mathrm{price}.(\overline{\mathrm{card}} \oplus \overline{\mathrm{cash}})$
- Seller =
  $\mathrm{bag}.\overline{\mathrm{price}}.(\mathrm{card} + \mathrm{cash}) + \mathrm{belt}.\overline{\mathrm{price}}.\mathrm{cash}$
- 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 = $\overline{\text{bag}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}}) + \overline{\text{belt}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}})$
- Seller = $\text{bag}.\overline{\text{price}}.(\text{card} + \text{cash}) + \text{belt}.\overline{\text{price}}.\text{cash}$
- 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' =
  $[] \prec \overline{\text{bag}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}}) + \overline{\text{belt}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}})$

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

Sample computation

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

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

Sample computation

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

- Seller =
  $[] \prec \mathrm{bag}.\overline{\mathrm{price}}.(\mathrm{card} + \mathrm{cash}) + \mathrm{belt}.\overline{\mathrm{price}}.\mathrm{cash}$
  - $\mathrm{bag}.\overline{\mathrm{price}}.(\mathrm{card} + \mathrm{cash}) \prec \overline{\mathrm{price}}.\mathrm{cash}$
  - $\mathrm{bag}.\overline{\mathrm{price}}.(\mathrm{card} + \mathrm{cash}) : \circ \prec \mathrm{cash}$

Sample computation

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

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

Sample computation

- Buyer' =
  $\overline{\text{bag}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}}) : \circ \prec \overline{\text{card}}$

- Seller =
  $\text{bag}.\overline{\text{price}}.(\text{card} + \text{cash}) : \circ \prec \text{cash}$

Sample computation

- Buyer’ =

  $\overline{\text{bag}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}}) : \circ \prec \overline{\text{card}}$

  ‣ $\overline{\text{bag}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}}) \prec \circ$

- Seller =

  $\text{bag}.\overline{\text{price}}.(\text{card} + \text{cash}) : \circ \prec \text{cash}$

  ‣ $\text{bag}.\overline{\text{price}}.(\text{card} + \text{cash}) \prec \circ$

Sample computation

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

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

Sample computation

- Buyer’ =
  $[] \prec \overline{\text{bag}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}})$

- Seller =
  $[] \prec \text{bag}.\overline{\text{price}}.(\text{card} + \text{cash})$

Sample computation

- Buyer’ =
  $[] \prec \overline{\text{bag}}.\text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}})$
- $\circ \prec \text{price}.(\overline{\text{card}} \oplus \overline{\text{cash}})$

- Seller =
  $[] \prec \text{bag}.\overline{\text{price}}.(\text{card} + \text{cash})$
- $\circ \prec \overline{\text{price}}.(\text{card} + \text{cash})$

Sample computation

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

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

Sample computation

- Buyer’ =

  $\circ : \circ \prec \overline{\text{card}} \oplus \overline{\text{cash}}$

- Seller =

  $\circ : \circ \prec \text{card} + \text{cash}$

Sample computation

- Buyer' =

  $\circ : \circ \prec \overline{\text{card}} \oplus \overline{\text{cash}}$

  ▸ $\circ : \circ \prec \overline{\text{card}}$

- Seller =

  $\circ : \circ \prec \text{card} + \text{cash}$

Sample computation

- Buyer' =

  $\circ : \circ \prec \overline{\text{card}} \oplus \overline{\text{cash}}$

  ▸ $\circ : \circ \prec \overline{\text{card}}$

  ▸ $\circ : \circ : \circ \prec 1$

- Seller =

  $\circ : \circ \prec \text{card} + \text{cash}$

  ▸ $\circ : \circ : \text{cash} \prec 1$

Map of the talk

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

Compliance

- The compliance relation $h \prec \sigma \dashv\!\!\| \; k \prec \rho$ holds iff $h \prec \sigma \,\|\, k \prec \rho \rightarrow^{*} h' \prec \sigma' \,\|\, k' \prec \rho' \not\rightarrow$ 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

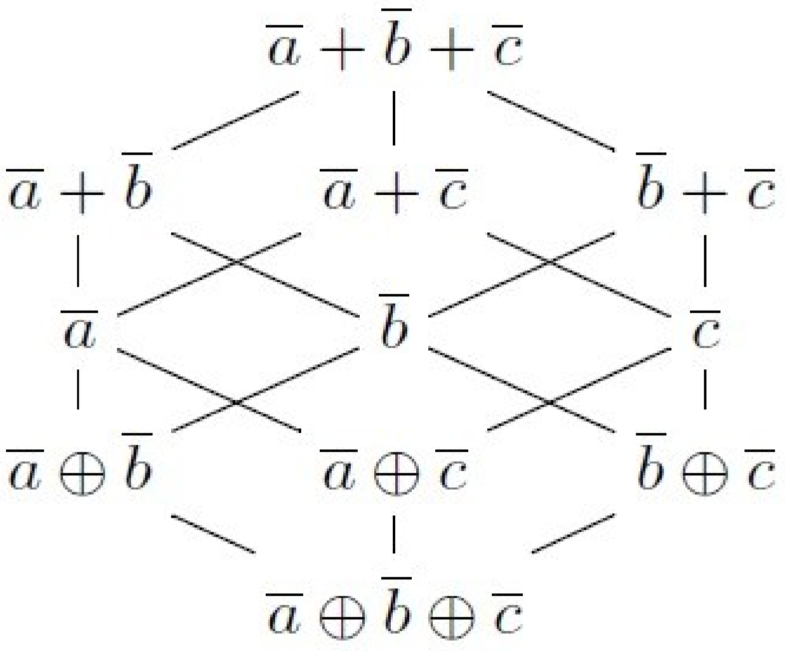
Compliance: results

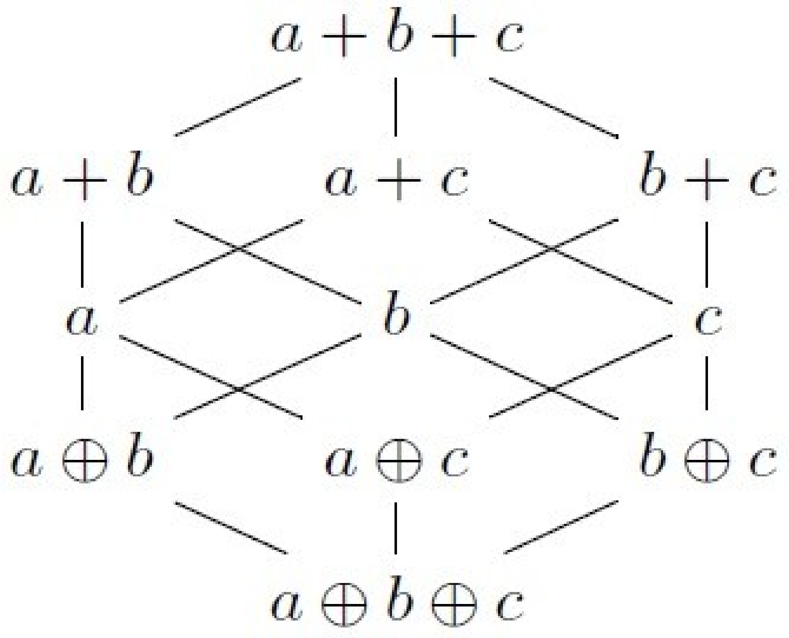
- Compliance is decidable even for contracts with recursion
- The complexity is $O(n^5)$
  - Straightforward algorithm is exponential
- The algorithm extends in a non trivial way the one for subtyping of recursive arrow and product types from Pierce

Subcontract relation

- Subcontract relation for servers:
  $\rho \preccurlyeq_s \rho'$ iff for each client σ. $\sigma \dashv\!\vert \rho$ implies $\sigma \dashv\!\vert \rho'$
  - ρ has more clients than ρ'
- Subcontract relation for clients is dual:
  $\sigma \preccurlyeq_c \sigma'$ iff for each server ρ. $\sigma \dashv\!\vert \rho$ implies $\sigma' \dashv\!\vert \rho$
- The two subcontract relations are partial orders
- The dual $\overline{\sigma}$ of a client contract σ is the minimum server compliant with σ

Subcontract relation: example


$\overline{a}+\overline{b}+\overline{c}$
$\overline{a}+\overline{b}$
$\overline{a}+\overline{c}$
$\overline{b}+\overline{c}$
$\overline{a}$
$\overline{b}$
$\overline{c}$
$\overline{a}\oplus\overline{b}$
$\overline{a}\oplus\overline{c}$
$\overline{b}\oplus\overline{c}$
$\overline{a}\oplus\overline{b}\oplus\overline{c}$



Duality has a simple syntactic characterization

1	↔	1
$\Sigma_{i\in I}\ a_i.\sigma_i$	↔	$\bigoplus_{i\in I}\ \overline{a}_i.\sigma_i$
$\Sigma_{i\in I}\ \overline{a}_i.\sigma_i$	↔	$\bigoplus_{i\in I}\ a_i.\sigma_i$
X	↔	X
$\text{rec } X.\sigma$	↔	$\text{rec } X.\sigma$

Subcontract relation: results

- Subcontract relation for servers and for clients are related:
  $\rho \preccurlyeq_{s} \rho'$ iff $\overline{\rho'} \preccurlyeq_{c} \overline{\rho}$
- Subcontract relation and compliance are related:
  $\rho \preccurlyeq_{s} \rho'$ iff $\overline{\rho} \dashv\!| \; \rho'$
- 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

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