Certified Management of Financial Contracts

**Patrick Bahr**  Jost Berthold  Martin Elsman

University of Copenhagen, Department of Computer Science (DIKU)

26th Nordic Workshop on Programming Theory, 2014
Introduction

What are financial contracts?

- stipulate future transactions between different parties
- have time constraints
- may depend on stock prices, exchange rates etc.

Example (Foreign Exchange Option)

At any time within the next 90 days, party X may decide to buy USD 100 from party Y, for a fixed rate $r$ of Danish Kroner.
Introduction

What are financial contracts?

- stipulate future transactions between different parties
- have time constraints
- may depend on stock prices, exchange rates etc.

Example (Foreign Exchange Option)

At any time within the next 90 days, party X may decide to buy USD 100 from party Y, for a fixed rate $r$ of Danish Kroner.
Introduction

What are financial contracts?

- stipulate future transactions between different parties
- have time constraints
- may depend on stock prices, exchange rates etc.

Example (Foreign Exchange Option)

At any time within the next 90 days, party X may decide to buy USD 100 from party Y, for a fixed rate $r$ of Danish Kroner.

Goals

- Express such contracts in a formal language
- Symbolic manipulation and analysis of such contracts.
What are financial contracts?

- stipulate future transactions between different parties
- have time constraints
- may depend on stock prices, exchange rates etc.

Example (Foreign Exchange Option)

At any time within the next 90 days, party X may decide to buy USD 100 from party Y, for a fixed rate $r$ of Danish Kroner.

Goals

- Express such contracts in a formal language
- Symbolic manipulation and analysis of such contracts.
- Formally verified!
Contract Language Goals in Detail

- **Compositionality.**
  Contracts are time-relative \(\Rightarrow\) straightforward compositionality

- **Multi-party.**
  Specify obligations and opportunities for multiple parties, (which opens up the possibility for specifying portfolios)

- **Contract management.**
  Contracts can be managed and symbolically evolved; a contract gradually reduces to the empty contract.

- **Contract utilities (symbolic).**
  Contracts can be analysed in a variety of ways

- **Contract pricing (numerical, staged).**
  Code for payoff can be generated from contracts (input to a stochastic pricing engine)
Example

Contract in natural language

- At any time within the next 90 days,
- party X may decide to
- buy USD 100 from party Y,
- for a fixed rate $r$ of Danish Kroner.
Example

Contract in natural language

- At any time within the next 90 days,
- party X may decide to
- buy USD 100 from party Y,
- for a fixed rate $r$ of Danish Kroner.

Translation into contract language

\[
\text{if } (\text{obs}_B(X, 0), 90, \text{trade}, \text{zero})
\]

where \( \text{trade} = \text{scale}(100, \text{both} (\text{transfer}(Y, X, \text{USD}), \text{pay})) \)

\( \text{pay} = \text{scale}(r, \text{transfer}(X, Y, \text{DKK})) \)
Contributions

- **Denotational semantics** based on cash-flows
- **Reduction semantics** (sound and complete)
- Correctness proofs for common contract analyses and transformations
- **Formalised** in the Coq theorem prover
- **Certified implementation** via code extraction
An Overview of the Contract Language

Core Calculus of Contracts

\[
\begin{align*}
\text{zero} &: \text{Contr} \\
\text{transfer} &: \text{Party} \times \text{Party} \times \text{Currency} \rightarrow \text{Contr} \\
\text{both} &: \text{Contr} \times \text{Contr} \rightarrow \text{Contr} \\
\text{scale} &: \text{Expr}_{\mathbb{R}} \times \text{Contr} \rightarrow \text{Contr} \\
\text{translate} &: \mathbb{N} \times \text{Contr} \rightarrow \text{Contr} \\
\text{if} &: \text{Expr}_{\mathbb{B}} \times \mathbb{N} \times \text{Contr} \times \text{Contr} \rightarrow \text{Contr}
\end{align*}
\]
An Overview of the Contract Language

Core Calculus of Contracts

\[
\begin{align*}
\text{zero} & : \text{Contr} \\
\text{transfer} & : \text{Party} \times \text{Party} \times \text{Currency} \rightarrow \text{Contr} \\
\text{both} & : \text{Contr} \times \text{Contr} \rightarrow \text{Contr} \\
\text{scale} & : \text{Expr}_R \times \text{Contr} \rightarrow \text{Contr} \\
\text{translate} & : \mathbb{N} \times \text{Contr} \rightarrow \text{Contr} \\
\text{if} & : \text{Expr}_B \times \mathbb{N} \times \text{Contr} \times \text{Contr} \rightarrow \text{Contr}
\end{align*}
\]

Expression Language

\text{Expr}_R, \text{Expr}_B: \text{real-valued resp. Boolean-valued expressions.}
An Overview of the Contract Language

Core Calculus of Contracts

\[
\begin{align*}
\text{zero} & : \text{Contr} \\
\text{transfer} & : \text{Party} \times \text{Party} \times \text{Currency} \rightarrow \text{Contr} \\
\text{both} & : \text{Contr} \times \text{Contr} \rightarrow \text{Contr} \\
\text{scale} & : \text{Expr}_R \times \text{Contr} \rightarrow \text{Contr} \\
\text{translate} & : \mathbb{N} \times \text{Contr} \rightarrow \text{Contr} \\
\text{if} & : \text{Expr}_B \times \mathbb{N} \times \text{Contr} \times \text{Contr} \rightarrow \text{Contr}
\end{align*}
\]

Expression Language

\[\text{Expr}_R, \text{Expr}_B: \text{real-valued resp. Boolean-valued expressions.}\]

\[
\begin{align*}
\text{obs}_\alpha & : \text{Label}_\alpha \times \mathbb{Z} \rightarrow \text{Expr}_\alpha \\
\text{acc}_\alpha & : (\text{Expr}_\alpha \rightarrow \text{Expr}_\alpha) \times \mathbb{N} \times \text{Expr}_\alpha \rightarrow \text{Expr}_\alpha
\end{align*}
\]
Example: Asian Option

\[\text{translate}(90, \text{if}(\text{obs}_B(X, 0), 0, \text{trade}, \text{zero}))\]

where \( \text{trade} = \text{scale}(100, \text{both}(\text{transfer}(Y, X, \text{USD}), \text{pay})) \)

\( \text{pay} = \text{scale}(\text{rate}, \text{transfer}(X, Y, \text{DKK})) \)

\( \text{rate} = \frac{1}{30} \cdot \text{acc}(\lambda r.r + \text{obs}_R(\text{FX USD/DKK}, 0), 30, 0) \)
Denotational Semantics

The semantics of a contract is given by the cash-flow it stipulates.

\[ C \mathbin{[\cdot]} : \text{Contr} \rightarrow \text{CashFlow} \]
Denotational Semantics

The semantics of a contract is given by the cash-flow it stipulates.

\[ C \mathcal{[\cdot]} : \text{Contr} \rightarrow \text{CashFlow} \]

\[ \text{CashFlow} = \mathbb{N} \rightarrow \text{Transactions} \]
\[ \text{Transactions} = \text{Party} \times \text{Party} \times \text{Currency} \rightarrow \mathbb{R} \]
The semantics of a contract is given by the cash-flow it stipulates.

\[ C \mathbf{[\cdot]} : \text{Contr} \times \text{Env} \rightarrow \text{CashFlow} \]
\[ \text{Env} = \text{Label} \times \mathbb{Z} \rightarrow \mathbb{B} \cup \mathbb{R} \]

\[ \text{CashFlow} = \mathbb{N} \rightarrow \text{Transactions} \]
\[ \text{Transactions} = \text{Party} \times \text{Party} \times \text{Currency} \rightarrow \mathbb{R} \]
Denotational Semantics

The semantics of a contract is given by the cash-flow it stipulates.

\[ C[\cdot] : \text{Contr} \times \text{Env} \rightarrow \text{CashFlow} \]

\[ \text{Env} = \text{Label}_\alpha \times \mathbb{Z} \rightarrow \alpha \]

\[ \text{CashFlow} = \mathbb{N} \rightarrow \text{Transactions} \]

\[ \text{Transactions} = \text{Party} \times \text{Party} \times \text{Currency} \rightarrow \mathbb{R} \]
Contract Analyses

Examples

- contract dependencies
- contract causality
- contract horizon
Contract Analyses

Examples

- contract dependencies
- contract causality
- contract horizon

Semantics vs. Syntax

- these analyses have precise semantic definition
- they cannot be effectively computed
- we provide sound approximations, e.g. type system
Contract Transformations

**Contract equivalences**
When can we replace a sub-contract with another one, without changing the semantics of the contract?

**Reduction semantics**
What does the contract look like after \( n \) days have passed?

**Contract Specialisation**
What does the contract look like after we learned the actual value of some observables?
Contract Equivalences

\[
\begin{align*}
\text{translate}(d, \text{zero}) & \simeq \text{zero} \\
\text{scale}(r, \text{zero}) & \simeq \text{zero} \\
\text{scale}(0, c) & \simeq \text{zero} \\
\text{both}(c, \text{zero}) & \simeq c \\
\text{scale}(s_1, \text{scale}(s_2, c)) & \simeq \text{scale}(s_1 \cdot s_2, c) \\
\text{translate}(d_1, \text{translate}(d_2, c)) & \simeq \text{translate}(d_1 + d_2, c) \\
\text{translate}(d, \text{both}(c_1, c_2)) & \simeq \text{both}(\text{translate}(d, c_1), \text{translate}(d, c_2)) \\
\text{scale}(x, \text{both}(c_1, c_2)) & \simeq \text{both}(\text{scale}(x, c_1), \text{scale}(x, c_2)) \\
\text{translate}(d, \text{scale}(s, c)) & \simeq \text{scale}(s/d, \text{translate}(d, c)) \\
\text{translate}(d, \text{if}(b, e, c_1, c_2)) & \simeq \\
& \quad \text{if}(b/d, e, \text{translate}(d, c_1), \text{translate}(d, c_2)) \\
\text{both}(\text{scale}(x, \text{transfer}(a, b, c)), \text{scale}(y, \text{transfer}(a, b, c))) & \simeq \text{scale}(x + y, \text{transfer}(a, b, c))
\end{align*}
\]
Reduction Semantics

\[ c \xrightarrow{\tau} \rho \ c' \]
Reduction Semantics

\[ c \xrightarrow{\tau} \rho \ c' \]

\[ \text{transfer}(p_1, p_2, c) \xrightarrow{\tau_{p_1, p_2, c}} \rho \ \text{zero} \]
Reduction Semantics

$\quad c \xrightarrow{\tau} \rho \quad c'$

\[
\text{transfer}(p_1, p_2, c) \xrightarrow{\tau_{p_1, p_2, c}} \rho \quad \text{zero}
\]

\[
\text{scale}(e, c) \xrightarrow{\frac{v}{1}} \rho \quad \text{scale}(e/-1, c')
\]
Reduction Semantics

\[
c \xrightarrow{\tau} \rho \ c' 
\]

\[
\text{transfer}(p_1, p_2, c) \xrightarrow{\tau_{p_1, p_2, c}} \rho \ \text{zero}
\]

\[
c \xrightarrow{\tau} \rho \ c' \quad \mathcal{E} \left[ e \right]_{\rho} = v
\]

\[
\text{scale}(e, c) \xrightarrow{v \ast \tau} \rho \ \text{scale}(e / -1, c')
\]

\[\vdots\]
Reduction Semantics

\[ c \xrightarrow{\tau} \rho c' \]

\[ \text{transfer}(p_1, p_2, c) \xrightarrow{\tau_{p_1, p_2}, c} \rho \text{ zero} \]

\[ \text{scale}(e, c) \xrightarrow{v \ast \tau} \rho \text{ scale}(e/-1, c') \]

Theorem (Reduction semantics correctness)

(i) If \( c \xrightarrow{\tau} \rho c' \), then
   (a) \( C[c]_\rho(0) = \tau \), and
   (b) \( C[c]_\rho(i+1) = C[c']_{\rho/1}(i) \) for all \( i \in \mathbb{N} \).

(ii) If \( C[c]_\rho(0) = \tau \), then there is a unique \( c' \) with \( c \xrightarrow{\tau} \rho c' \).
Code Extraction

Coq formalisation

- Denotational & reduction semantics
- Meta-theory of contracts (causality, monotonicity, ...)
- Definition of contract transformations and analyses
- Correctness proofs
Code Extraction

Coq formalisation

- Denotational & reduction semantics
- Meta-theory of contracts (causality, monotonicity, ...)
- Definition of contract transformations and analyses
- Correctness proofs
Code Extraction

Coq formalisation

- Denotational & reduction semantics
- Meta-theory of contracts (causality, monotonicity, ...)
- Definition of contract transformations and analyses
- Correctness proofs

Extraction of executable Haskell code

- efficient Haskell implementation
- embedded domain-specific language for contracts
- contract analyses and contract management
Future Work

- improve code extraction
- advanced analyses and transformations (e.g. scenario generation and “zooming”)
- combine this work with numerical methods
Conclusion

The code is available from

http://j.mp/contractDSL

including

▸ full Coq proofs
▸ code extraction
▸ Prototype Haskell implementation
▸ example contracts
▸ technical report with all details