Certified Management of Financial Contracts

Patrick Bahr
DIKU
paba@diku.dk

joint work with
Jost Berthold & Martin Elsman

10th December, 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.
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.
- Formal verification
Contract Language Goals in Detail

- **Compositionality.**
  Contracts are time-relative ⇒ facilitates 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

if $\text{obs}(X \text{ exercises option})$ within 90
then $100 \times (\text{USD}(Y \rightarrow X) & r \times \text{DKK}(X \rightarrow Y))$
else $\emptyset$
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

∅ empty contract with no obligations

\( a(p_1 \rightarrow p_2) \) \( p_1 \) has to transfer one unit of \( a \) to \( p_2 \)

\( c_1 \& c_2 \) conjunction of \( c_1 \) and \( c_2 \)

\( e \times c \) multiply all obligations in \( c \) by \( e \)

\( d \uparrow c \) shift \( c \) into the future by \( d \) days

let \( x = e \) in \( c \) observe today’s value of \( e \) at any time (via \( x \))
An Overview of the Contract Language

∅ empty contract with no obligations

\( a(p_1 \rightarrow p_2) \) \( p_1 \) has to transfer one unit of \( a \) to \( p_2 \)

\( c_1 & c_2 \) conjunction of \( c_1 \) and \( c_2 \)

\( e \times c \) multiply all obligations in \( c \) by \( e \)

\( d \uparrow c \) shift \( c \) into the future by \( d \) days

let \( x = e \) in \( c \) observe today’s value of \( e \) at any time (via \( x \))

if \( e \) within \( d \) then \( c_1 \) else \( c_2 \)

- behave like \( c_1 \) as soon as \( e \) becomes true
- if \( e \) does not become true within \( d \) days behave like \( c_2 \)
An Overview of the Contract Language

∅ empty contract with no obligations

\( a(p_1 \rightarrow p_2) \) \( p_1 \) has to transfer one unit of \( a \) to \( p_2 \)

\( c_1 \& c_2 \) conjunction of \( c_1 \) and \( c_2 \)

\( e \times c \) multiply all obligations in \( c \) by \( e \)

\( d \uparrow c \) shift \( c \) into the future by \( d \) days

let \( x = e \) in \( c \) observe today’s value of \( e \) at any time (via \( x \))

if \( e \) within \( d \) then \( c_1 \) else \( c_2 \)

▶ behave like \( c_1 \) as soon as \( e \) becomes true

▶ if \( e \) does not become true within \( d \) days behave like \( c_2 \)

Expression Language

Real-valued and Boolean-valued expressions, extended by

\( \text{obs}(l, d) \) observe the value of \( l \) at time \( d \)

\( \text{acc}(f, d, e) \) accumulation over the last \( d \) days
Example: Asian Option

\[ 90 \uparrow \textbf{if } obs(X \text{ exercises option}) \textbf{ within } 0 \]
\[ \text{then } 100 \times (USD(Y \rightarrow X) \& (rate \times DKK(X \rightarrow Y))) \]
\[ \text{else } \emptyset \]

where

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

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

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

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

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

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

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

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

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

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

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

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

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

Examples

- contract dependencies
- contract causality
- contract horizon
Contract Analyses

Examples

- contract dependencies
- contract causality
- contract horizon

\[ \text{obs}(FX \ USD/\ DKK, 1) \times DKK(X \rightarrow Y) \]
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 Causality

Refined Types

- $e : \text{Expr}^t_{\alpha}$: value of $e$ available at time $t$ (or later)
- $c : \text{Contr}^t$: no obligations strictly before $t$
Contract Causality

Refined Types

- \( e : \text{Expr}^t_\alpha \)  
  value of \( e \) available at time \( t \) (or later)
- \( c : \text{Contr}^t \)  
  no obligations strictly before \( t \)

Typing Rules

\[
\begin{align*}
t_1, t_2 &\in \mathbb{Z} & l &\in \text{Label}_\alpha & t_1 &\leq t_2 \\
\Gamma &\vdash obs(l, t_1) : \text{Expr}^{t_2}_\alpha \\
p_1, p_2 &\in \text{Party} & a &\in \text{Asset} \\
\vdash a(p_1 \rightarrow p_2) : \text{Contr}^0
\end{align*}
\]
Contract Causality

Refined Types

- $e : \text{Expr}^t_{\alpha}$ value of $e$ available at time $t$ (or later)
- $c : \text{Contr}^t$ no obligations strictly before $t$

Typing Rules

\[
\frac{t_1, t_2 \in \mathbb{Z} \quad l \in \text{Label}_{\alpha} \quad t_1 \leq t_2}{\Gamma \vdash \text{obs}(l, t_1) : \text{Expr}^{t_2}_{\alpha}}
\]

\[
\frac{p_1, p_2 \in \text{Party} \quad a \in \text{Asset}}{\vdash a(p_1 \rightarrow p_2) : \text{Contr}^0}
\]

\[
\frac{\vdash e : \text{Expr}^t_{R} \quad \vdash c : \text{Contr}^t}{\vdash e \times c : \text{Contr}^t}
\]

\[
\frac{d \in \mathbb{N}}{\vdash d \uparrow c : \text{Contr}^{t+d}}
\]
Contract Causality

Refined Types

- $e : \text{Expr}^{t}_{\alpha}$, value of $e$ available at time $t$ (or later)
- $c : \text{Contr}^{t}$, no obligations strictly before $t$

Typing Rules

\[
\frac{t_1, t_2 \in \mathbb{Z} \quad l \in \text{Label}_{\alpha} \quad t_1 \leq t_2} {\Gamma \vdash \text{obs}(l, t_1) : \text{Expr}^{t_2}_{\alpha}}
\]

\[
\frac{p_1, p_2 \in \text{Party} \quad a \in \text{Asset}} {\vdash a(p_1 \rightarrow p_2) : \text{Contr}^{0}}
\]

\[
\frac{\vdash e : \text{Expr}^{t}_{R} \quad \vdash c : \text{Contr}^{t}} {\vdash e \times c : \text{Contr}^{t}}
\]

\[
\frac{d \in \mathbb{N}} {\vdash d \uparrow c : \text{Contr}^{t+d}}
\]
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

\[ e_1 \times (e_2 \times c) \simeq (e_1 \cdot e_2) \times c \]
\[ d_1 \uparrow (d_2 \uparrow c) \simeq (d_1 + d_2) \uparrow c \]
\[ d \uparrow (c_1 \& c_2) \simeq (d \uparrow c_1) \& (d \uparrow c_2) \]
\[ e \times (c_1 \& c_2) \simeq (e \times c_1) \& (e \times c_2) \]
\[ d \uparrow (e \times c) \simeq (d \uparrow e) \times (d \uparrow c) \]
\[ d \uparrow (\text{if } b \text{ within } e \text{ then } c_1 \text{ else } c_2) \simeq \]
\[ \text{if } d \uparrow b \text{ within } e \text{ then } d \uparrow c_1 \text{ else } d \uparrow c_2 \]
\[ (e_1 \times a(p_1 \to p_2)) \& (e_2 \times a(p_1 \to p_2)) \simeq (e_1 + e_2) \times a(p_1 \to p_2) \]
Reduction Semantics

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

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

\[
\begin{align*}
\text{a}(p_1 \rightarrow p_2) & \xrightarrow{\tau_{a,p_1,p_2}} \rho \emptyset \\
\end{align*}
\]
Reduction Semantics

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

\[
\begin{align*}
\text{a}(p_1 \rightarrow p_2) & \xrightarrow{\tau_{a,p_1,p_2}} \rho \emptyset \\
\end{align*}
\]

\[
\begin{align*}
E \left[ e \right]_\rho & = v \\
E \left[ e \right]_\rho & = v \times c \xrightarrow{\rho} (-1 \uparrow e) \times c'
\end{align*}
\]
Reduction Semantics

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

\[ \begin{align*}
\text{If } \mathbf{a}(p_1 \rightarrow p_2) & \xrightarrow{\tau_a, p_1, p_2} \rho \emptyset \\
\text{then } \mathbf{e} \times \mathbf{c} & \xrightarrow{\rho} (-1 \uparrow \mathbf{e}) \times \mathbf{c}' \\
\text{and } \mathbf{E}[\mathbf{e}]_{\rho} & = \nu
\end{align*} \]
**Theorem (Reduction semantics correctness)**

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

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

\[ a(p_1 \rightarrow p_2) \xrightarrow{\tau_{a, p_1, p_2}} \rho \emptyset \]

\[ e \times c \xrightarrow{v*\tau} \rho (-1 \uparrow e) \times c' \]

\[ \vdots \]
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
- further analyses and transformations (e.g. scenario generation and “zooming”)
- combine this work with numerical methods