Swiss Verification Day 2024

EPFL

FV-ACTUS

Formally Verifiable **ACTUS**

Mark A. Greenslade

- Head of R&D @ Casper Association
- Member of Central Bank Research Association
- mark@casper.network

Quinn Dougherty

- Formalization engineer of R&D @ Casper Association
- Fan of declarative technologies
- quinn@casper.network

casper

Part 1: ACTUS

Part 2: ACTUS -> ZK -> DLT

Part 3: ZK Proofs

Part 4: Formal verification strategies (*speculative*)

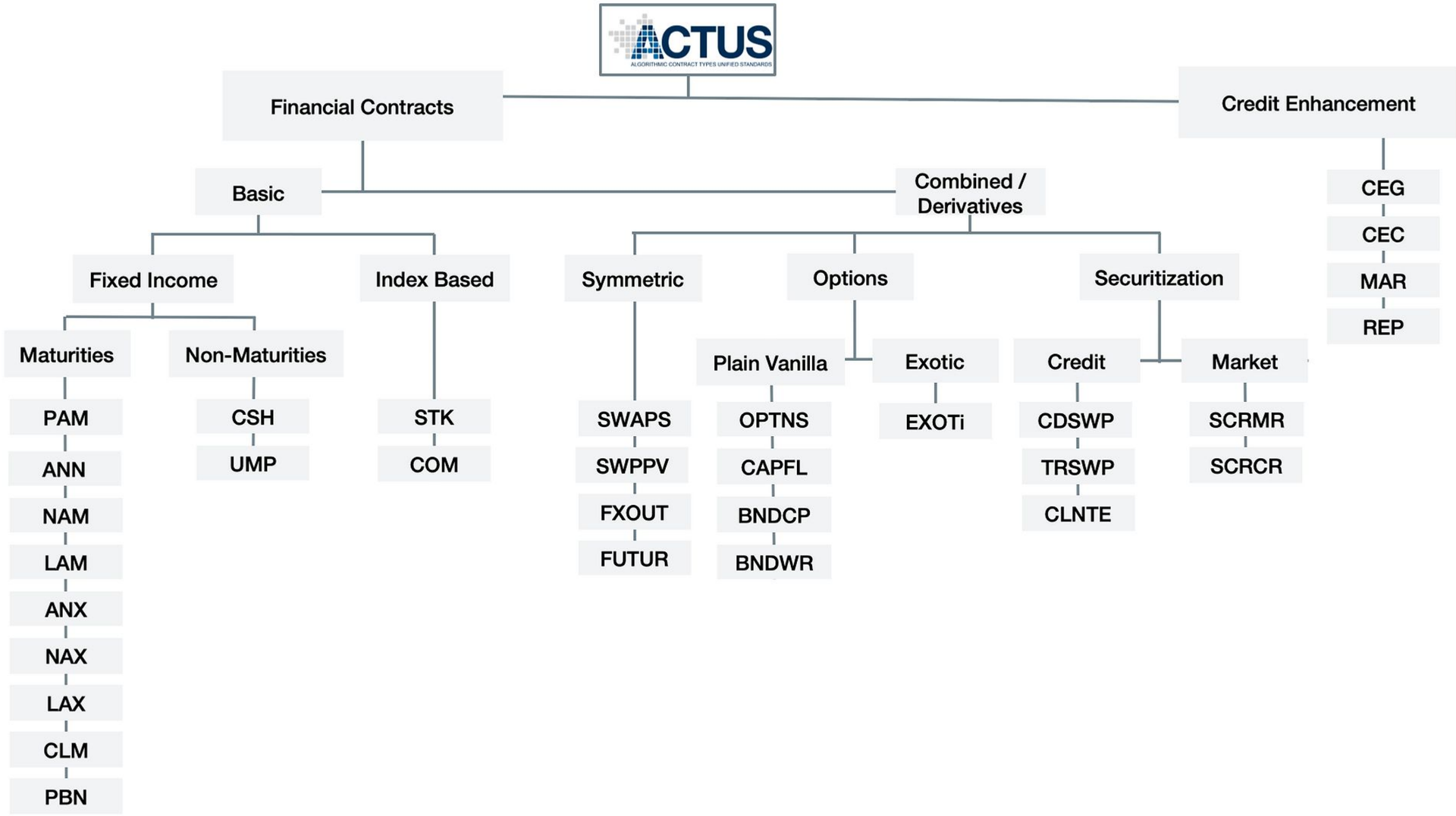
Part 1: ACTUS

An Emerging Financial Standard

actusfrf.org

ACTUS

Taxonomy



Specification example: PAM

7.1. **PAM: Principal At Maturity.**

PAM: Contract Schedule

Event	Schedule	Comments
AD	$\vec{t}^{AD} = (t_0, t_1, ..., t_n)$	With $t_i, i = 1, 2, ...$ a custom input
IED	$t^{IED} = \texttt{IED}$	
MD	$t^{MD} = \mathbf{Md}_{t_0}$	
PP	$\vec{t}^{PP} = \begin{cases} \varnothing & \text{if} \quad \texttt{PPEF} = \text{'N'} \\ (\vec{u}, \vec{v}) & \text{else} \end{cases}$ where $\vec{u} = S(s, \texttt{OPCL}, T^{MD})$ $\vec{v} = O^{ev}(\texttt{CID}, \texttt{PP}, t)$	with $s = \begin{cases} \varnothing & \text{if} \quad \texttt{OPANX} = \varnothing \wedge \texttt{OPCL} = \varnothing \\ \texttt{IED} + \texttt{OPCL} & \text{else if} \quad \texttt{OPANX} = \varnothing \\ \texttt{OPANX} & \text{else} \end{cases}$
PY	$\vec{t}^{PY} = \begin{cases} \varnothing & \text{if} \quad \texttt{PYTP} = \text{'O'} \\ \vec{t}^{PP} & \text{else} \end{cases}$	
FP	$\vec{t}^{FP} = \begin{cases} \varnothing & \text{if} \quad \texttt{FER} = \varnothing \vee \texttt{FER} = 0 \\ S(s, \texttt{FECL}, T^{MD}) & \text{else} \end{cases}$	with $s = \begin{cases} \varnothing & \text{if} \quad \texttt{FEANX} = \varnothing \wedge \texttt{FECL} = \varnothing \\ \texttt{IED} + \texttt{FECL} & \text{else if} \quad \texttt{FEANX} = \varnothing \\ \texttt{FEANX} & \text{else} \end{cases}$
PRD	$t^{PRD} = \texttt{PRD}$	
TD	$t^{TD} = \texttt{TD}$	

Algorithms

- Types
  - Utility functions
  - Payoff functions
  - State transition functions
- Inputs
  - Heterogeneous termsets
  - Termsets are composable
- Output
  - Event Sequence (1..N)
    - Equivalent to cash flows
  - Homogeneous

Part 2: ACTUS -> ZK -> DLT

Verifiable Financial Contracts

ACTUS + ZK + DLT

ACTUS

(Counter Parties, Term Set, Algorithm, Cash Flows)

Cryptographic Proofs

(Signatures, Attestations, Fingerprints, ZK-Proofs)

DLT

(Smart Contract)

ACTUS + ZK + DLT

VFC Tokenisation

Minting

(Identifiers, Direction, Counter Parties, Units, Metadata)

⇩

DLT

(Smart Contract)

⇧

Servicing

(Auditors, Rating, Regulators, Markets)

ACTUS + ZK + DLT

ACTUS Cash Flow

(Timestamp, Direction, Amount, Denomination, Obligor)

Payments Engine

(Verify, Calculate, Open, Close, Default, Notify)

DLT

(Smart Contract)

Part 3: Zero Knowledge Proofs

Computational Integrity

Zero Knowledge

ZK-Proofs

$f(x,w) \rightarrow \{True,False\}$

Properties

Succinct
Sound
Computationally Expensive
Cheap to Verify

Elements

Arithmetic Circuit
Finite Field
Constraint System
Polynomial Commitment

Developers

Virtual Machines
E-DSLs
Rollups
Applications

Part 4: Formal verification strategies (*speculative*)

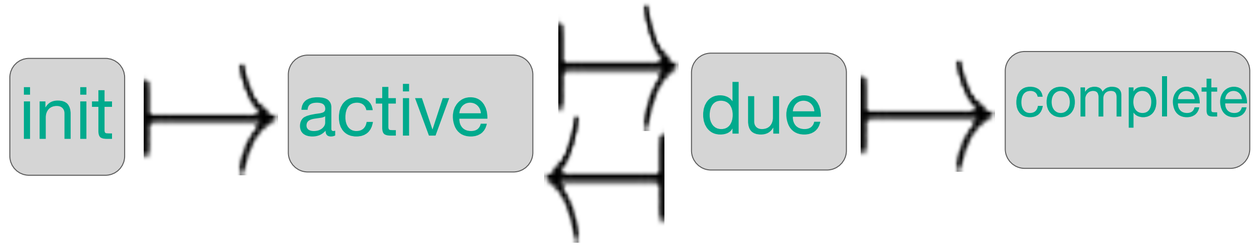
Towards a new era of *executing* and *auditing* finance

Formal verification strategies

Reactive System

Principal at maturity (PAM) *(simplified)*: The atomic ACTUS instrument PAM is a simple loan. It provides principal quantity, interest rate, and repayment schedule.

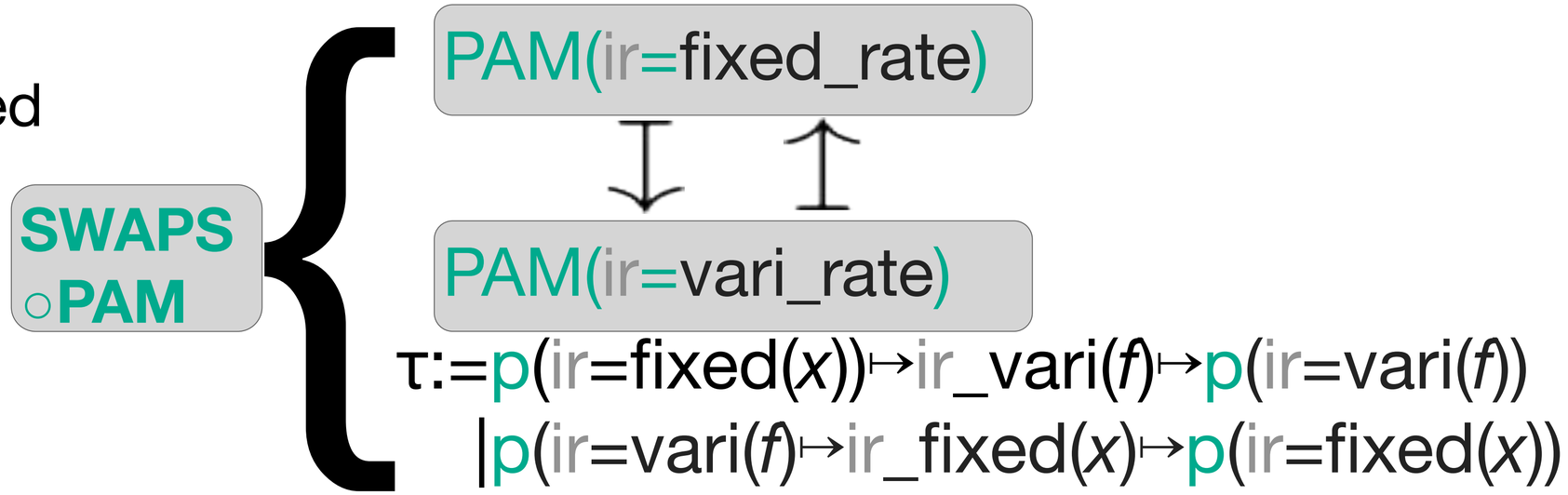
PAM
- princ
- ir
- schd



$$\tau := \text{active} \mapsto t(\text{schd}) \mapsto \text{due} \mid \text{due} \mapsto x \mapsto \text{if } \text{princ} \times (1+\text{ir}) - x = 0 \text{ then complete else active}$$

Formal verification strategies

loan with swap:
ACTUS instruments principal at maturity (**PAM**) and swap (**SWAPS**) can be piped together to make a loan of swappable interest rate, so **SWAPS∘PAM** would have a transition element mapping a fixed rate to a variable rate and another vice versa

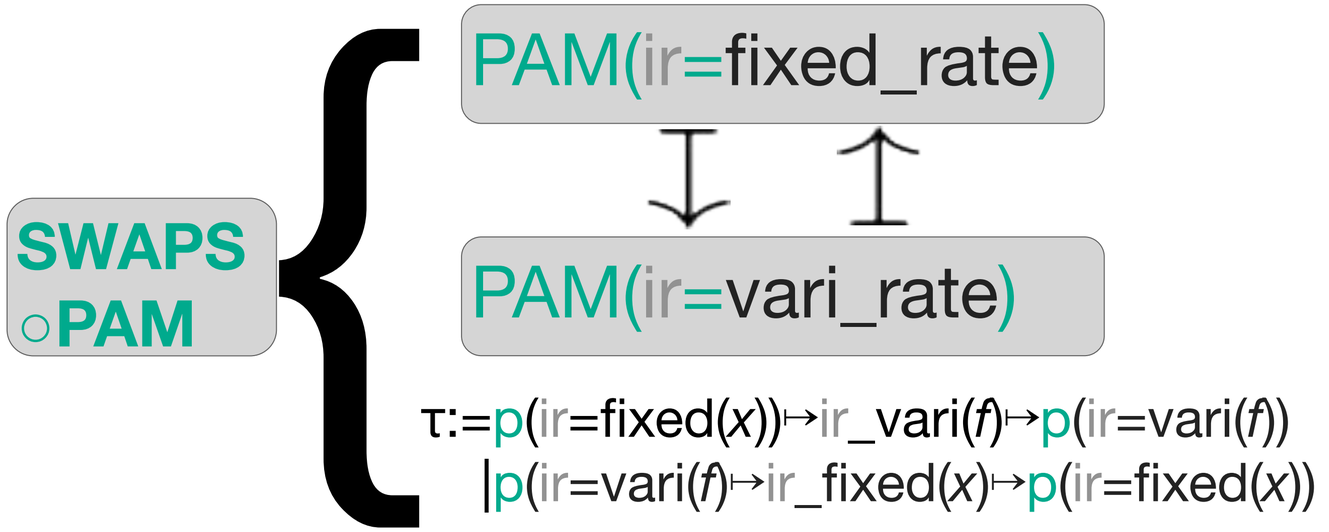


Formal verification strategies

Reactive System

SWAPS○PAM:
The **composed** contract could respond to **external market events** (like changes in the benchmark interest rate) by adjusting the interest payments according to the terms of the swap contract.

This dynamic adjustment is similar to how a reactive system responds to changes in its environment, making the financial contract adaptive and responsive to real-world conditions.



Formal verification strategies

Other Directions?

- **Abstract interpretation**
  - An arithmetization process (for the ZK stack) may go smoother in an abstract domain
    - May need to lose numeric precision to make proof jobs small enough, since ZK provers are expensive in time and space
  - We would like static analysis to let us map assurances bought in the abstract domain back to the concrete domain
- **Hoare logic**
  - The proof of each triple ought to be the same oneliner (induction), if the state transition semantics are done properly
  - But writing pre and post conditions could be a clarifying QA procedure for intricately composed instruments
- **Something in the coalgebra weeds**
  - Coalgebras can be a design principle for state transition systems like ACTUS contracts
  - Perhaps algebra mages have some deductive tricks
- **Other ways to leverage SMT solvers (besides reactive system)**
- **Linear temporal logic**

Swiss Verification Day 2024

EPFL

ZK-ACTUS

Verifiable Financial Contracts

Swiss Verification Day 2024

EPFL

ZK-ACTUS

Verifiable Financial Contracts

mark@casper.network | quinn@casper.network

Mark Greenslade & Quinn Dougherty Jan ‘24