Pacioli: a PROLOG system for financial report validation

Miguel Calejo (mc@logicalcontracts.com)
Charles Hoffman (Charles.Hoffman@me.com)

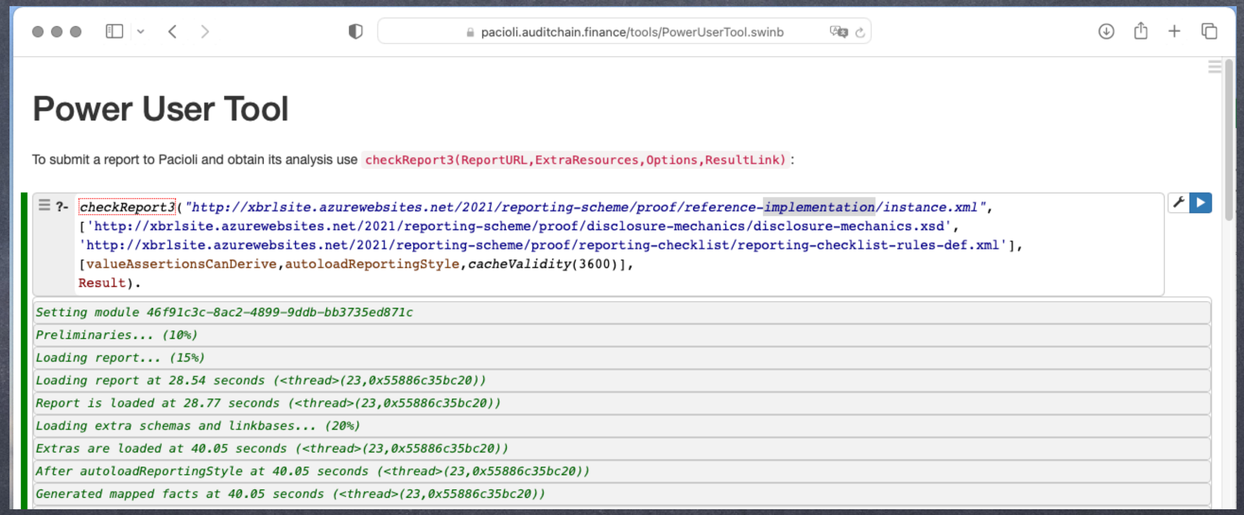
Developed for auditchain.finance

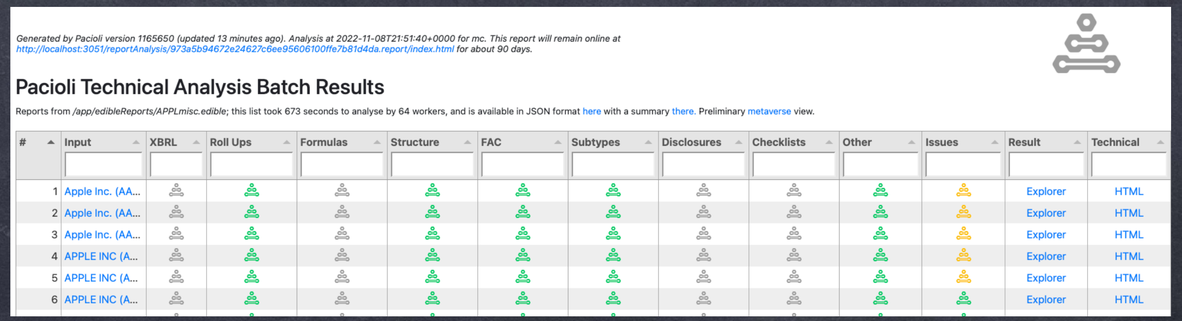
Context

- "eXtensible Business Reporting language" is used in 60+ jurisdictions in 5 continents
- a common format, XBRL is output by many different accounting systems worldwide
  - Example: human-readable, raw XBRL, PROLOG:-), human again
- Needs validation/auditing, prior and after public filings
  - Rules galore: concept mappings, roll ups, structure, controls... may be pre-included or injected into the report

Pacioli at work

- Submit example via Pacioli notebook ; result
- Example batch reports:
  - Apple, last decade
  - DOW 30
  - Fortune 100
- Web3 app





Under the hood

- Some Prolog code:
  - loadXBRL_
  - evaluateFormula
  - block detection: rollUp
  - typeSubtypeViolation
  - reportsHTML
- The report model
  - Prolog representation
  - JSON subsets

```
% Aspects argument refers to the whole formula (implicit) aspects
% AspectModels refers to all aspect models, one per fact variable; notice that the unit aspect contains unit IDs, not measure terms
% PostFilter contains filters that need to run at the end, because they depend on nonlocal variables
% SupportFact will be bound to a factKey(Concept,Context,Unit,Why) supporting this binding; Why reports concept mappings and fact origin
bindFormulaVar(R,ID,Simple,CommonContext,VarName,Fallback,Conditions,Value,AspectModels,[DimensionsA,EntityA,PeriodA,UnitA,ConceptA],LinkRole,SupportFact,(Goal,ContextExcluder,Filter),(ContextExcluder,PostFilter)) :-
    % too strong, would fail some aggregations, so we comment it out:
    % ((member(Condition,Conditions), Condition\=fallback(_)) -> true ; (complain(R,error,bindFormulaVar(ID,VarName),"Unfiltered variable, I refuse to handle it"), fail)),
    (Simple==true -> Context=CommonContext ; true),  %commented out because it explodes the use of fallbacks, ergo bogus assertion failures etc; cf. MC email to CH and JM May 20, 2021
    Fact = fact_precise(R,Concept,Context,Unit,_Precision,Decimals,Value__,Why),
    SupportFact = factKey(Concept,Context,Unit,Why),
    Goal = (
        ((Fact, ContextFilter) , % this was restricting subsequent contexts to this fact:  *->
            (
                (Value__==null->Value_=Fallback;Value_=Value__),
                contextWithDimensionsFor(Concept,Simple,R,LinkRole,Context,Entity,Segments_,Period), Segments_=Segments,
                ContextExcluder=true,
                length(Segments,SegmentsSize), constrainSegmentsSize(AspectModels,SegmentsSize)
            ) ;
                (
                    (Fallback\==null -> Value_=Fallback ; IsFormula==true -> Value_=null),
                    Why=failed, DimensionsA=Segments, PeriodA=Period, UnitA=Unit, ContextFilter,
                    % if we have a fallbackValue, we do NOT accept a context+unit for it for which we DO have a real value
                    % findunique(Acontext+Aunit,fact_precise(R,Concept,Acontext,Aunit,_,_Decimals,_Value,_Why),FactContextsUnits), ContextExcluder=( \+ member(Context+Unit,FactContextsUnits))
                    Concept=Prefix:Concept_,
                    % the following is tested for bindings first; and because in the first fact of a formula some vars may stiull be unbound... this test is repeated at the end, after filters:
                    ContextExcluder = ( \+ (ground(Concept_+Context+Unit), explicitContextUnit(Concept_,Prefix,Context,Unit,R)))
                )),
                    % ???? should leave this, unit etc unbound??
        %contextWithDimensionsFor(Concept,Simple,R,LinkRole,Context,Entity,Segments_,Period), % any network; formulas are above networks!
        %compatibleSegments(Segments_,Segments),
        applyDecimals(Decimals,Value_,Value)
        %, writeln(Concept/Context/LinkRole/Value)
    ),
    Entity=EntityA, % TODO: supporting other filters involving this aspect
    %TODO: transform value as per precision
    [Segments,Entity,Period,Unit,Concept] = AM, memberchk(VarName-AM,AspectModels),
    (select(concept(Concept),Conditions,C1) -> true ; (Conditions=C1, ConceptA=Concept )), % no concept: potential explosion?
    ( \+ member(explicitDimension(_,_),C1) -> DimensionsA=Segments ; true),
    ( \+ member(instantDuration(_,_),C1) -> PeriodA=Period ; true),
    ( \+ member(unit(_),C1) -> UnitA=Unit ; true),
    ( limitedContexts(R,LC) -> ContextFilter=member(Context,LC); ContextFilter=true),
    ( myFormula(R,_,_,ID,formula,_,_,_) -> IsFormula=true ; IsFormula=false),
    bindFormulaVarConditions(R,ID,C1,AspectModels,AM,LinkRole,Filter,PostFilter).
```

Query report models

- addBatchedReportModels(‘https://auditchain.infura-ipfs.io/ipfs/QmSYSWjjwypXeYhWCFrYwNX6jqVq3nXYAQ1dL3behL2dDG/reports.json').

- What is the biggest inconsistency reported?

```
DeltaGoal = (
  ruleOutcome(R,i,RuleID,RuleInstanceID,SupportingFacts,Delta),
  Delta\==null, Delta_ is abs(Delta),
  \+ member(factKey(_,_,_,failed),SupportingFacts) ,
  \+ member(factKey(_,_,_,facInitialZeros/_), SupportingFacts),
  memberchk(factKey(_,Context,_,_),SupportingFacts),
  context(R,Context,_,_Segments,Period)
  ),
findunique(Delta_-What-RuleID-Context-Period, (DeltaGoal,pacioliReport(R,_,What,_)),LL_),
reverse(LL_,Sorted),
forall(member(Delta-What-RuleID-Context-Period,Sorted),
  format("~D:\tin ~a, rule ~w , ~w (context ~a)~n",[round(Delta),What,RuleID,Period,Context])).
```

```
330,267,000,000:        in American International Group, Inc. (AIG) 10-K for FY, 2021, rule cal
128,670,000,000:        in WALT DISNEY CO/ (DIS) 10-K for FY, 2021, rule calculation(http://cor
99,487,000,000: in Prudential Financial, Inc. (PRU) 10-K for FY, 2021, rule calculation(http://
99,487,000,000: in Prudential Financial, Inc. (PRU) 10-K for FY, 2021, rule FAC_CONSISTENCY_16
78,582,000,000: in WELLS FARGO & COMPANY/MN (WFC) 10-K for FY, 2021, rule calculation(http://ww
78,542,000,000: in BERKSHIRE HATHAWAY INC (BRK.A) 10-K for FY, 2021, rule calculation(http://ww
C_0001067983_20210101_20211231)
```

Getting real: blockchain nodes

AUDITCHAIN

WEB3 INFRASTRUCTURE

AUDITCHAIN LUCA SUITE

BUSINESS AND INTELLIGENCE
CONTROL COMPOSITION AND
VALIDATION

EXTERNAL VALIDATION

PACIOLI NODE OPERATORS

Composition UI
Ontologies, Semantics

Pacioli Logic & Reasoning Engine

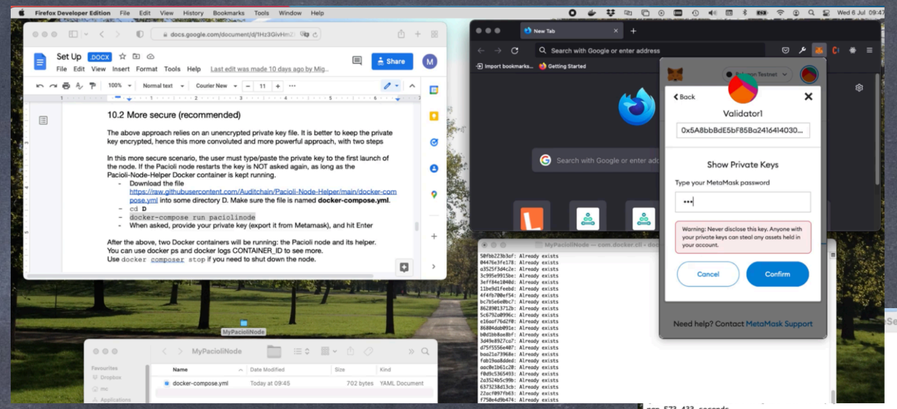
Ontology and Semantic Control Verification

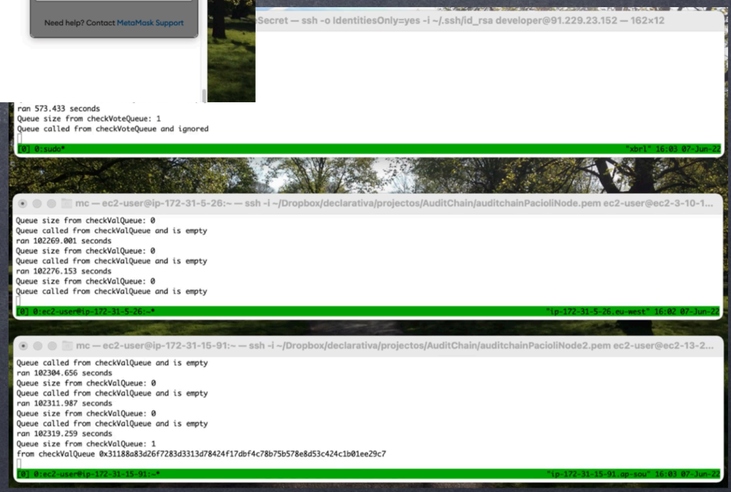
LEARN MORE

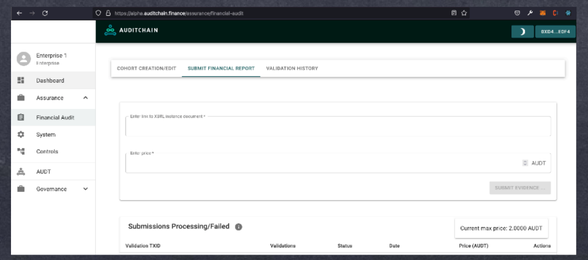
External Validation of Process Control NFTs

The Pacioli node network

- One node (Docker): Web3 Jscript driving a local Pacioli
- Launching a node
- Some nodes running
- Inside a node
- Submitting a report for validation







Conclusion

- Auditchain's Pacioli
  - Pacioli intersects logic programming with XBRL and SBRM
    - new rule types; Logical English; etc.
    - related tools - Luca, etc
  - Spring 2022: AUDT token was listed
  - Beta testing (Polygon Mumbai), for a year now; Mainnet launch expected Q1 2023
- PROLOG rocks at 50!
  - Logical variable, DCG-based UI generation, web stack, fabric for DSLs, multi language integration, libraries, portability... and logic too :-) declarative AND procedural
  - Powering "niche" apps that matter
  - Prolog developers needed! pls email Miguel if you're hireable