Pacioli: a PROLOG system for financial report validation

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

https://auditchain.finance

1 Introduction

Financial information is reported by public companies to regulators worldwide using a standard rich structured data format, the "Extensible Business Reporting Language", or XBRL[1]. An XBRL report typically comprises two pieces:

- A single XML (*or alternatively "inline XBRL", iXBRL[2])* file with instance data – financial facts contextualized with dates and hypercube dimensions – which depends on:
- Background ontological information, in the form of a "Discoverable Taxonomy Set" – potentially dozens of files with data element schema and hypercube definitions, hierarchical presentation directives, assertions - a tree graph of XML schema and linkbase resources published on the web.

Pacioli[3] provides:

- Loading of XBRL reports and their Discoverable Taxonomy Sets into a self-contained Prolog model.
- Prolog-based XBRL formula processor, adding auditing and explanation capabilities.
- Combination of a report model with user alterations of formulas and facts, as well as additional linkbases.
- Fact mapping and derivation, for "ontology mapping" all reports into the same comparable representation
- Detection of higher-level "blocks" of information, beyond XBRL, such as roll ups, roll forwards, adjustments, disclosures, etc.
- Processing of XBRL and extra-XBRL rules for report validation, including disclosure mechanics and checklists, model structure, and "Fundamental Accounting Concepts" rules; this is described by the Seattle Method[4] of processing XBRL-based financial reports;
- All rule outcomes are persisted to the Pacioli model on IPFS[5], anchored to a Merkle-like cryptographic hash of all the above ingredients, thus ensuring immutability and reproducibility for posterity
- Multiple report rendering interfaces

So for example, given Apple's 10K for 2021 inline XBRL filing[6], Pacioli produces an analysis[7]; one information block of which is shown in the pivot table below:

[1] https://www.xbrl.org/introduction/

[2] https://www.xbrl.org/the-standard/what/ixbrl/

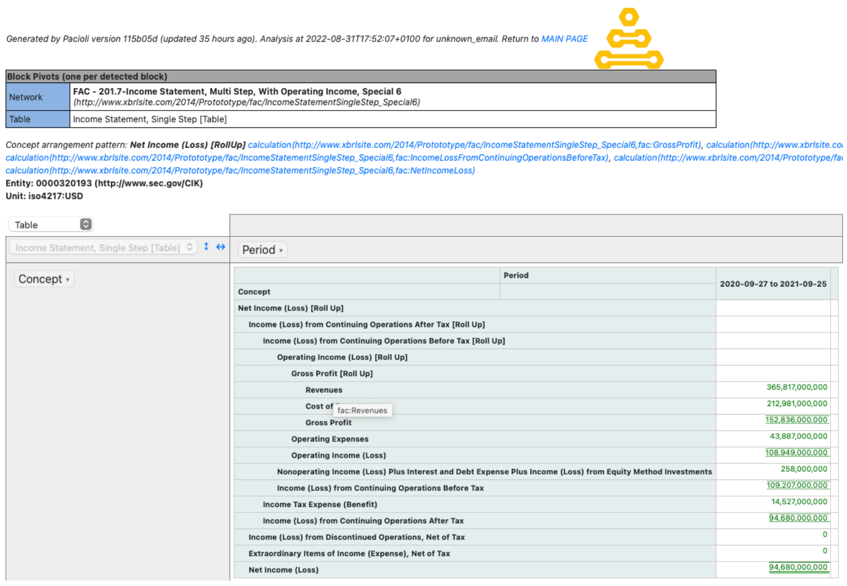
[3] https://docs.auditchain.finance/auditchain-protocol/pacioli-logic-and-rules-engine

[4] Charles Hoffman, CPA, Seattle Method, http://xbrlsite.com/seattlemethod/SeattleMethod.pdf

[5] https://ipfs.io , the "InterPlanetary File System", a decentralized, redundant, robust file storage service

[6] https://www.sec.gov/Archives/edgar/data/320193/000032019321000105/aapl-20210925.htm

[7] https://auditchain.infura-ipfs.io/ipfs/QmSuMTNG1W98U3xTsJRX2cs1LxKQqGKqM9iq2w1HhsaCZB/



Generated by Pacioli version 115b05d (updated 35 hours ago). Analysis at 2022-08-31T17:52:07+0100 for unknown_email. Return to MAIN PAGE

Block Pivots (one per detected block)	
Network	**FAC - 201.7-Income Statement, Multi Step, With Operating Income, Special 6** *(http://www.xbrlsite.com/2014/Protototype/fac/IncomeStatementSingleStep_Special6)*
Table	Income Statement, Single Step [Table]

Concept arrangement pattern: ***Net Income (Loss) [RollUp]*** *calculation(http://www.xbrlsite.com/2014/Protototype/fac/IncomeStatementSingleStep_Special6,fac:GrossProfit), calculation(http://www.xbrlsite.co*
calculation(http://www.xbrlsite.com/2014/Protototype/fac/IncomeStatementSingleStep_Special6,fac:IncomeLossFromContinuingOperationsBeforeTax), calculation(http://www.xbrlsite.com/2014/Protototype/fa
calculation(http://www.xbrlsite.com/2014/Protototype/fac/IncomeStatementSingleStep_Special6,fac:NetIncomeLoss)
Entity: 0000320193 (http://www.sec.gov/CIK)
Unit: iso4217:USD

Concept	Period: 2020-09-27 to 2021-09-25
Net Income (Loss) [Roll Up]	
Income (Loss) from Continuing Operations After Tax [Roll Up]	
Income (Loss) from Continuing Operations Before Tax [Roll Up]	
Operating Income (Loss) [Roll Up]	
Gross Profit [Roll Up]	
Revenues	365,817,000,000
Cost of	212,981,000,000
Gross Profit	152,836,000,000
Operating Expenses	43,887,000,000
Operating Income (Loss)	108,949,000,000
Nonoperating Income (Loss) Plus Interest and Debt Expense Plus Income (Loss) from Equity Method Investments	258,000,000
Income (Loss) from Continuing Operations Before Tax	109,207,000,000
Income Tax Expense (Benefit)	14,527,000,000
Income (Loss) from Continuing Operations After Tax	94,680,000,000
Income (Loss) from Discontinued Operations, Net of Tax	0
Extraordinary Items of Income (Expense), Net of Tax	0
Net Income (Loss)	94,680,000,000

If you follow the link in the previous footnote and navigate to MAIN PAGE / Derivations Graph, you can see how the facts above were derived from the filed data.

PROLOG lovers can add a suffix to the URL of any Pacioli report and obtain its PacioliModel[8], in this case 36k PROLOG facts for the Apple report and analysis. Searching in there for the fac:'Revenue' fact in the pivot table above, you'll find that it was not filed, but actually mapped from an us-gaap[9] concept that was reported:

```
mappedFact(…,
    fac:'Revenues',
    'i55e5364a9af5491886caee077afe8d44_D20200927-20210925',
    usd,
    null,-6,
    365817000000,
    'http://accounting.auditchain.finance/2022/fac/Rules_Mapping/COMID-BSC-CF1-ISM-IEMIB-OILY-SPEC6_mapping-definition.xml'
+
        ('us-gaap':'RevenueFromContractWithCustomerExcludingAssessedTax') +
        reported
    ).
```

For more examples, see the Pacioli batch report[10] for recent Dow Jones top 30 company filings.

2 System Architecture

Pacioli has been available since early 2021 as a web application at http://pacioli.auditchain.finance to support debugging, rule development and training. Select users (developers, accountants) interact with SWISH[11] notebooks[12]: they submit financial report URLs and obtain report validation analyses from Pacioli, as self-contained HTML mini sites generated by SWI-Prolog's termerised HTML[13], using Javascript frameworks[14] for browser client-side data rendering. The report analysis output includes machine-readable (Prolog and JSON) files, all stored on IPFS:

[8] For the above example, https://auditchain.infura-ipfs.io/ipfs/QmSuMTNG1W98U3xTsJRX2cs1LxKQqGKqM9iq2w1HhsaCZB/ReportAndModel.pl.gzip

[9] https://www.cfainstitute.org/en/advocacy/issues/gaap#sort=%40pubbrowsedate%20descending

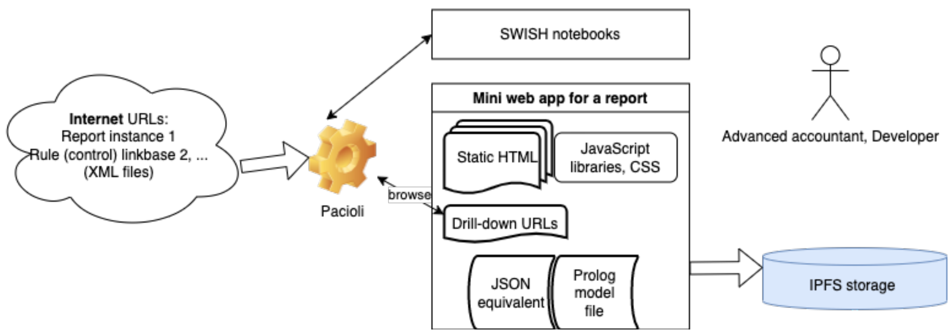
[10] https://auditchain.infura-ipfs.io/ipfs/QmaATb3njmXgbbZVuUPuJweukyHNk2WbxGVJCSEUgqRt3o/

[11] https://github.com/SWI-Prolog/swish

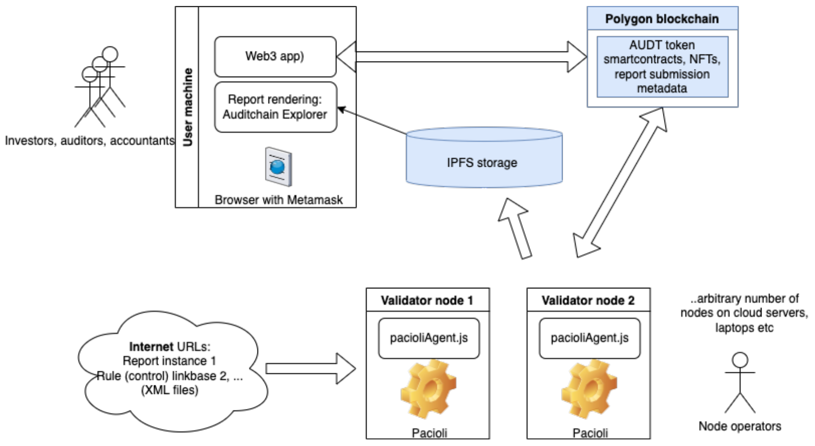
[12] For exemple https://pacioli.auditchain.finance/tools/PowerUserTool.swinb

[13] https://github.com/Anniepoo/swiplwebtut/blob/master/web.adoc#html1-and-termerized-html

[14] Namely https://pivottable.js.org/examples/ and http://tabulator.info, in addition to SWISH-generated Graphviz graphs


SWISH notebooks
Mini web app for a report
Static HTML
JavaScript libraries, CSS
Drill-down URLs
JSON equivalent
Prolog model file
Internet URLs: Report instance 1 Rule (control) linkbase 2, ... (XML files)
Pacioli
browse
Advanced accountant, Developer
IPFS storage

But its main use case is embedded in "Pacioli nodes", constituting the decentralized Auditchain validator network:


Investors, auditors, accountants
User machine
Web3 app)
Report rendering: Auditchain Explorer
Browser with Metamask
Polygon blockchain
AUDT token smartcontracts, NFTs, report submission metadata
IPFS storage
Internet URLs: Report instance 1 Rule (control) linkbase 2, ... (XML files)
Validator node 1
pacioliAgent.js
Pacioli
Validator node 2
pacioliAgent.js
Pacioli
..arbitrary number of nodes on cloud servers, laptops etc
Node operators

Pacioli coordination is performed by an Auditchain nodejs agent, and most users interface via Auditchain's Web3 app. Financial report rules will be (in 2023) wrapped as Non Fungible Tokens. The AUDT token acts as currency for user fees, network operation rewards and the future rule NFTs marketplace.

Since late 2021 this network went through alpha and beta programs with a dozen validators across the planet, with (Polygon) mainnet deployment imminent.

3 Some PROLOG implementation aspects

Since April 2020, Pacioli development indulged us on a gourmet feast of PROLOG programming[15]:

- XBRL to PROLOG conversion is a perfect fit for SWI-Prolog's XML parser, with XBRL standard definitions mapping to PROLOG clauses handling the XBRL DOM representation.
- Excel to PROLOG conversion in order to load information into Pacioli which is then converted to XBRL.
- To save development time, detailed XBRL conformance tests are delegated to a reference XBRL (pre)processor[16], launched as a PROLOG subprocess.
- Powerful ad-hoc querying over the Pacioli model with PROLOG[17]
- XBRL formula evaluation, including variable binding, is done by the straight execution of a Prolog goal generated from the formula; XBRL search over report facts and multidimensional contexts in a XML document maps straight into PROLOG backtracking over their relational representation, with expressions evaluated by a simple PROLOG interpreter
- Simplified syntax for XBRL formulae resorting to Prolog operators, as opposed to the original XML linkbase
- Block detection via declarative clauses
- Report rendering: "termerised HTML"[18] galore; external Javascript frameworks are configured and embedded via DCG-based HTML generation

[15] As we write this the open sourcing of Pacioli is still under discussion, hence no code URL yet

[16] https://github.com/Arelle/Arelle, also used to extract XBRL from iXBRL

[17] Simulating the XULE expression language in PROLOG: https://pacioli.auditchain.finance/example/XULE.swinb

[18] https://github.com/Anniepoo/swiplwebtut/blob/master/web.adoc#html1-and-termerized-html

4 Significance – present and future

PROLOG application projects usually contain a heavy component of research: new language implementation techniques, new system tools, or even theoretical advances. On the contrary, projects built with mature, mainstream languages focus on application rather than infra structure development.

Pacioli is probably one of the first PROLOG projects focusing on application development driven from business requirements, as reflected in the core team: a senior professional PROLOG developer and a domain guru – no (present…) academics onboard☺ This would be impossible without the underlying maturity of SWI-Prolog and its libraries.

Up to now a dozen Pacioli instances, operated by different entities around the world, have already validated thousands of financial reports in different jurisdictions. As Auditchain deploys to mainnet (*production*) these numbers will increase drastically, propelled by a global need from small and medium investors to scrutiny public companies outside the walled gardens of the big accounting firms; building up a trustable decentralized repository of financial report analysis, with PROLOG validators behind it, attesting confidence in the trillions of dollars of reported finances all over the planet – documented in thousands of actionable PROLOG models, amenable to later cross-querying etc.

Arguably, one of the most significant PROLOG applications to date, both in scale and global social impact.

5 References

Please see footnotes

Acknowledgements to Jason Meyers and the Auditchain early investors, for betting Auditchain on Pacioli; to Jacinto Dávila for help with Pacioli and for his Auditchain Explorer UI; to Bogdan Fiedur for the blockchain agent and Auditchain smartcontracts; to Christopher Jastrzebski for all the support, and his Auditchain Web3 UI; to Fuad Begic for the complementary Luca report editor under development; to Andrew Noble and Dudley Gould for encouragement and suggestions.

To Jan Wielemaker and associates, for a software stack that finally made PROLOG feasible for business applications.

And finally, to Bob Kowalski: for his vision half century ago, and all the personal support in recent years.