An Ontology Design Pattern for Representing Temporal Indirection

Yulia Svetashova[1]

[1]Data Management Lab, Bloomberg, 3 Queen Victoria St, London EC4N 4TQ

Abstract

We present the Temporal Indirection Ontology Design Pattern, which models persistent identifiers whose referents evolve over time according to deterministic rules. The pattern introduces core abstractions that separate identifier persistence from time-dependent referent dynamics, enabling both stable referencing and precise historical reconstruction. This approach addresses a significant gap in existing temporal ontologies, which typically assume static identifier-referent relationships and therefore cannot capture scenarios where meaning evolves but referential continuity must be preserved.

We validate the pattern through implementations in financial derivatives (e.g., generic identifiers such as "CL1" for front-month oil futures) and air transportation (flight numbers with daily-changing assignments). Our contributions include: (i) a reusable pattern grounded in foundational ontology concepts, (ii) competency question coverage demonstrating support for resolution, audit, and analysis tasks, and (iii) cross-domain evidence of applicability.

Keywords

Ontology Design Pattern, Temporal Indirection, Persistent Identifiers, Temporal Modeling

1. Introduction

Many information systems rely on identifiers that remain stable while the entities they reference change systematically over time. This creates a fundamental challenge for knowledge representation: how can we preserve referential stability while accurately tracking what an identifier denotes at any given moment? Humans readily understand expressions like "the current president" or "the department head" as stable references to roles whose occupants change. Capturing this logic in an ontology, however, requires explicit modeling. We call this phenomenon *temporal indirection*—a situation where a persistent identifier maintains constant identity while the referent changes systematically according to deterministic rules.

Consider financial markets, where generic trading symbols like "CL1" serve as persistent identifiers for front-month crude oil futures contracts. This identifier provides a paradigmatic example of temporal indirection within financial information systems. A persistent identifier (CL1) serves as a stable referent while dynamically resolving to the front-month crude oil futures contract, thereby ensuring continuity across successive monthly contract rollovers. In January 2025, "CL1" refers to the February 2025 contract; when that contract approaches expiration, "CL1" automatically transitions to reference the March 2025 contract. Real-time market attributes—such as last trade price, trading volume, and settlement values—are bound to the current contract instance but exposed through the invariant identifier.

This design illustrates the separation of identifier persistence from referent volatility, a principle central to data ontology and financial knowledge representation. Operational implementations of this mechanism can be observed in public systems such as MarketWatch and TradingView, each of which provides continuous charting and contract metadata under the "CL1" identifier despite the underlying contract substitutions.

This pattern extends far beyond finance. Table 1 shows representative domains where temporal indirection proves essential. Air transportation employs stable flight numbers (e.g., "UA001") that map

16th Workshop on Ontology Design and Patterns (WOP 2025) at ISWC, November 02–06, 2025, Nara, Japan

ysvetashova@bloomberg.net (Y. Svetashova)
https://github.com/YuliaS (Y. Svetashova)
0000-0003-1807-107X (Y. Svetashova)

to different aircraft and crews daily. Healthcare systems track roles like “primary oncologist on duty” that persist while individual physicians rotate through shifts. In each case, the identifier provides continuity for reporting and analysis, while precise resolution determines the actual entity at any point in time.

Domain	Persistent Identifier	Changing Referent	Use Case
Healthcare	Primary Oncologist Ward 7	Individual physicians	Treatment continuity versus per shift malpractice attribution
Environmental	River Station 1 Primary	Physical sensors	Regulatory compliance versus sensors calibration and tracking
Digital Services	Payment API Latest	Implementation versions	Client stability versus versions debugging and monitoring
Regulatory	Current GDPR	Specific rulings	Compliance continuity versus interpretation, guidance updates

Table 1
Representative examples of temporal indirection across domains

1.1. Limitations of Current Approaches

Existing temporal ontologies fail to capture this indirection pattern adequately. Standard approaches like PROV-O [1] focus on entity provenance but assume identifiers maintain fixed semantics. OWL-Time [2] provides temporal intervals but cannot model systematic referent evolution. The Financial Industry Business Ontology (FIBO) [3, 4], despite its 2,300+ classes, lacks mechanisms for rolling contract references. These limitations leave a critical gap: no existing pattern separates identifier persistence from referent dynamics while maintaining complete temporal traceability.

1.2. Our Contribution

We address this gap through the Temporal Indirection Ontology Design Pattern, which provides:

- A formal separation between persistent identifiers and their time-dependent referents,
- Explicit representation of transition rules and resolution contexts,
- Support for dual attribution (observations linked to both identifier and specific referent),
- Complete audit trails of all referent transitions.

Figure 1 illustrates the pattern’s core structure: a two-level system where stable identifiers connect to changing referents through time-dependent mappings.

1.3. Paper Organization

The remainder of this paper is structured as follows. Section 2 presents the Temporal Indirection Pattern’s abstract structure, beginning with core concepts and progressing through relationships, constraints, and competency questions. Section 3 demonstrates the pattern’s application to financial derivatives, showing how abstract classes map to domain concepts. Section 4 validates the pattern through comprehensive testing in both finance and aviation domains. Section 5 positions our contribution relative to existing temporal modeling approaches. Finally, Section 6 discusses implications and future directions.

2. The Temporal Indirection Pattern

This section presents the Temporal Indirection Ontology Design Pattern as a reusable solution for modeling persistent identifiers with evolving referents. We begin with the pattern’s conceptual foundations, then detail its structure, constraints, and validation criteria.

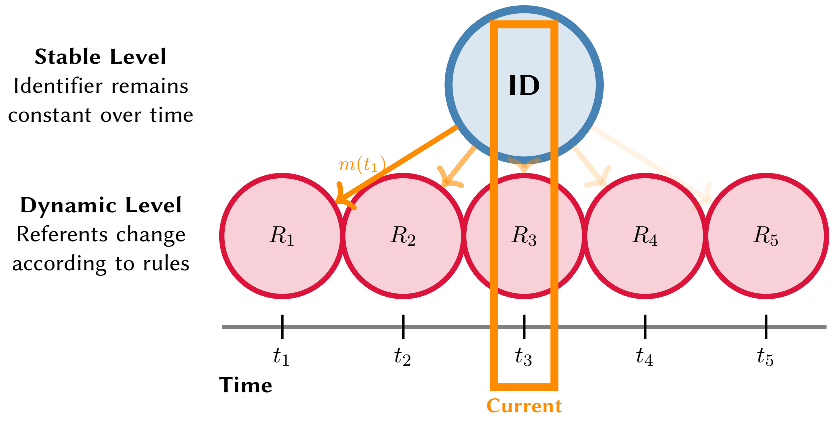

Figure 1: Temporal indirection model: The mapping function $m(t)$ determines which specific referent entity R is bound to the persistent identifier ID at any given time point t.

2.1. Conceptual Foundations

The pattern addresses four fundamental requirements that emerge from temporal indirection scenarios:

1. **Identifier Persistence**: Identifiers must maintain stable identity across time, independent of their referents.
2. **Deterministic Resolution**: At any time point, the mapping from identifier to referent must be unambiguous.
3. **Rule-Based Transitions**: Referent changes must follow explicit, domain-specific rules.
4. **Historical Traceability**: All past mappings must be preserved for reconstruction and audit.

To satisfy these requirements, the pattern introduces a layered architecture that separates concerns: persistent identifiers exist at a stable layer, referent entities exist at a dynamic layer, and temporal bindings connect the two layers according to resolution rules.

2.2. Core Components

The pattern can be used independently or aligned with foundational ontologies. It comprises seven core classes, each serving a specific role in the temporal indirection mechanism. Figure 2 illustrates their relationships.

2.2.1. Persistent Identifier and Referent Entity

A **Persistent Identifier** represents a stable reference that maintains consistent semantics for external systems. It exists independently of any particular referent and persists even during transition periods. Examples include trading symbols ("CL1"), flight numbers ("UA001"), or role designations ("Primary Oncologist, Ward 7"). When aligned with foundational ontologies, a persistent reference can be considered an information object (e.g., `dul:InformationObject`[1] in the DOLCE+DnS Ultralite (DUL) [5] ontology).

A **Referent Entity** is the actual entity that fulfills the persistent identifier at a given time. These entities have independent lifecycles: they exist before becoming associated with an identifier and continue after the association ends. In our financial example, specific futures contracts (e.g., "CLU25" for January 2025 delivery) serve as referent entities. In foundational ontology terms, these align with general entity concepts (e.g., `dul:Entity`).

[1]Throughout this paper, we use the following namespace prefixes:
`: <http://ontologydesignpatterns.org/cp/owl/temporalindirection.owl#>`,
`fd: <http://example.org/financial-derivatives#>`,
`avo: <http://example.org/aviation-operations#>`.
Standard prefixes (`rdf:`, `rdfs:`, `owl:`, `xsd:`, `dul:`) follow conventions as documented at https://prefix.cc/.

2.2.2. Temporal Binding

The **Temporal Binding** reifies the relationship between identifier and referent during a specific interval. Rather than using simple properties, this reification enables:

- Precise temporal boundaries through start and end times
- Tracking of transition events that initiate and terminate the binding
- Validation of temporal integrity (no gaps or overlaps)

Each binding connects exactly one persistent identifier to exactly one referent entity for a well-defined temporal interval[2]. Conceptually, it is a situation or context (cf. `dul:Situation`).

2.2.3. Transition Events and Rules

A **Transition Event** marks the moment when one binding ends and another begins. It captures:

- The outgoing referent (via `:changesFrom`)
- The incoming referent (via `:changesTo`)
- The precise timestamp of transition
- The rule that triggered the change (via `:triggeredBy`)

Transition Rules encode the domain logic governing when and how transitions occur. In financial markets, these might specify "roll to next month's contract on the 20th business day." In aviation, they might state "assign next available aircraft 2 hours before scheduled departure."

2.2.4. Resolution Context and Conventions

The **Resolution Context** provides the interpretive framework for determining active referents. It specifies which conventions apply, which rules are active, and how exceptions are handled. This concept parallels descriptive or normative elements in foundational ontologies (e.g., `dul:Description`) but functions independently as a rule specification.

Conventions represent domain-specific standards that govern temporal indirection. Examples include exchange trading calendars, hospital shift schedules, or software deployment policies.

Observations about entities require special handling to maintain both referential stability and specific attribution. An observation is linked to: 1) the persistent reference via `:observedEntity` (for continuity), 2) the specific referent entity that was active at observation time via `:observedReferentEntity` (for attribution), 3) observable properties that may apply to either the reference or the referent, 4) results that must be contextualized by both levels of indirection. This dual attribution enables queries that aggregate observations across all referents of a persistent reference while maintaining traceability to specific entities.

2.3. Structural Constraints

The pattern enforces several constraints to ensure temporal integrity:

Completeness: Every persistent identifier must have at least one temporal binding and exactly one resolution context. This prevents the creation of orphaned identifiers.

Determinism: Each temporal binding must connect to exactly one referent entity for a specific interval. This ensures unambiguous resolution.

Traceability: Every transition event must specify both source and target referents, along with the triggering rule. This enables complete audit trails.

[2]In our implementation, we did not explicitly reuse `time:Interval` or related properties from OWL-Time. Instead, we introduced a custom `:TemporalInterval` class with `:hasStartTime` and `:hasEndTime`. In future iterations, we may consider aligning with OWL-Time by reusing `time:hasBeginning`/`time:hasEnd` for intervals and `time:Instant` for event timestamps, to leverage existing standards and improve interoperability.

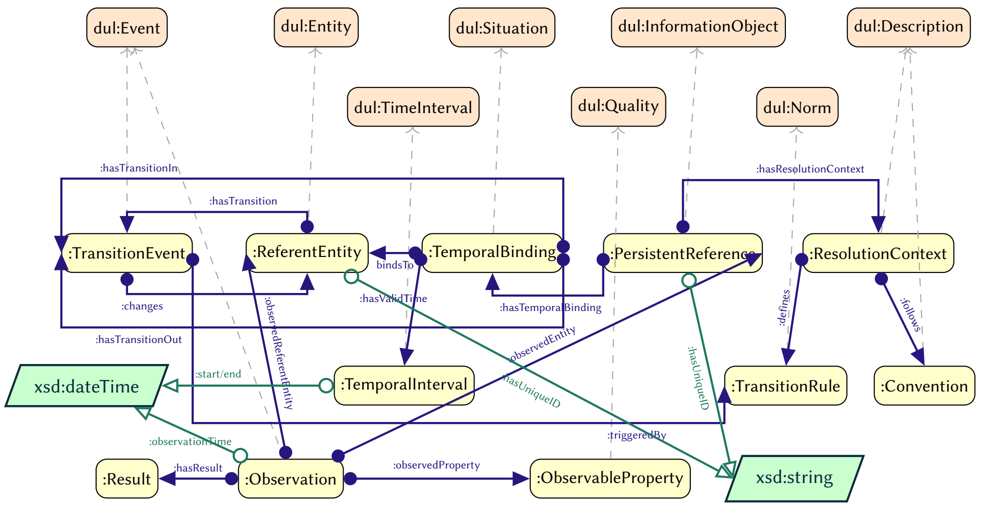

Figure 2: Temporal Indirection Pattern structure showing core classes (yellow) and their alignment with foundational ontology concepts (orange).

These constraints can be formalized in Description Logic as follows:

$$\begin{aligned}
\mathtt{PersistentReference} &\sqsubseteq \exists\mathtt{hasTemporalBinding}.\mathtt{TemporalBinding} \\
&\quad \sqcap (=1\,\mathtt{hasResolutionContext}.\top) \\
\mathtt{TemporalBinding} &\sqsubseteq (=1\,\mathtt{bindsTo}.\mathtt{ReferentEntity}) \\
&\quad \sqcap (=1\,\mathtt{hasValidTime}.\mathtt{TemporalInterval}) \\
\mathtt{TemporalBinding} &\sqsubseteq \exists\mathtt{hasTransitionIn}.\mathtt{TransitionEvent} \\
&\quad \sqcap \exists\mathtt{hasTransitionOut}.\mathtt{TransitionEvent} \\
\mathtt{TransitionEvent} &\sqsubseteq (=1\,\mathtt{changesFrom}.\mathtt{ReferentEntity}) \\
&\quad \sqcap (=1\,\mathtt{changesTo}.\mathtt{ReferentEntity}) \\
&\quad \sqcap (=1\,\mathtt{triggeredBy}.\mathtt{TransitionRule})
\end{aligned}$$

2.4. Competency Questions

To validate the pattern's completeness, we define eight competency questions that any implementation must answer:

ID	Type	Question
CQ1	Current Resolution	Which entity does a persistent identifier currently reference?
CQ2	Historical Resolution	Which entity was referenced at a past timestamp?
CQ3	Transition Timing	When did referent transitions occur?
CQ4	Attribution	What observations are associated with specific referents?
CQ5	Complete History	What is the full sequence of referents over time?
CQ6	Rule Provenance	Which rules triggered specific transitions?
CQ7	Temporal Integrity	Are there gaps or overlaps in temporal bindings?
CQ8	Context Reconstruction	Can complete context be rebuilt for any time point?

Table 2
Competency questions for pattern validation

These questions ensure the pattern supports both real-time operations (CQ1) and historical analysis (CQ2-5), while maintaining auditability (CQ6, CQ8) and data quality (CQ7).

3. Financial Derivatives Implementation

Having established the abstract pattern, we now demonstrate its application to financial derivatives, where rolling futures contracts exemplify temporal indirection. This section shows how each pattern component maps to financial concepts while preserving the pattern's semantic integrity.

3.1. Domain Context

In commodity futures markets, traders rely on standardized symbols known as *tickers* to identify contracts. These generic identifiers, such as "CL1" for the front-month crude oil contract, allow traders to reference rolling positions without specifying which particular delivery month currently holds that designation. This enables continuous market analysis: traders can chart "CL1" prices over years without manually switching between individual contracts as they expire.

The challenge lies in maintaining this referential stability while ensuring precise attribution. When a trader executes against "CL1," the actual contract (e.g., "CLG25" for February 2025 delivery) must be recorded for settlement. Similarly, historical analysis must reconstruct which specific contracts comprised a spread relationship at any point in time.

3.2. Pattern Specialization

Figure 3 illustrates how the pattern's abstract classes specialize for financial derivatives. Each core class maps to exactly one financial concept:

- `fd:GenericTicker` $\sqsubseteq$ `:PersistentReference` — stable market symbols
- `fd:FuturesContract` $\sqsubseteq$ `:ReferentEntity` — specific dated contracts
- `fd:ContractMapping` $\sqsubseteq$ `:TemporalBinding` — active ticker-contract relationships
- `fd:ContractRoll` $\sqsubseteq$ `:TransitionEvent` — monthly roll events
- `fd:RollSchedule` $\sqsubseteq$ `:ResolutionContext` — exchange-specific rules

This one-to-one mapping ensures, domain experts can understand the model while preserving the pattern's temporal reasoning capabilities.

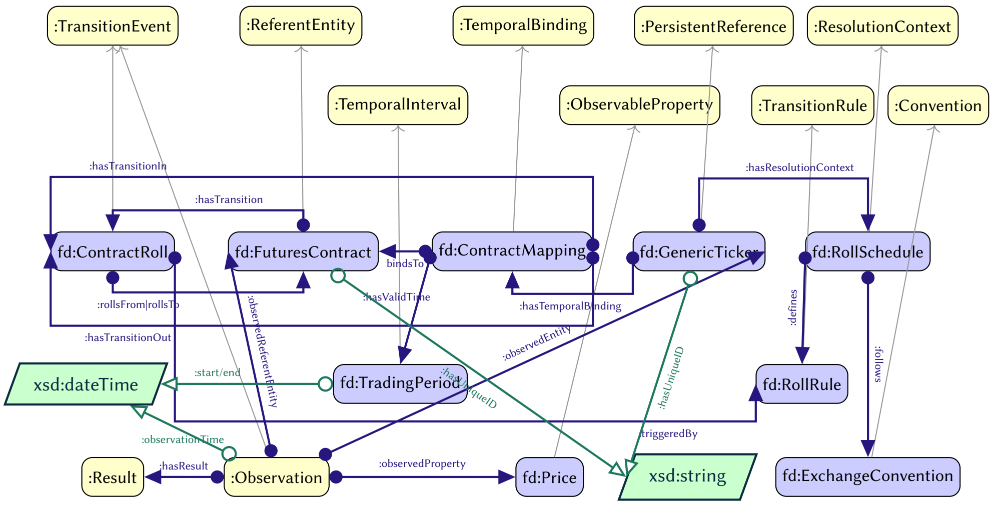

Figure 3: Financial derivatives specialization of the Temporal Indirection Pattern. Core pattern classes (yellow) are subclassed by financial concepts (blue).

3.3. Domain Extensions

Beyond core specializations, the financial implementation adds domain-specific elements:

Market Infrastructure: Classes for `fd:Commodity`, `fd:ContractMonth` provide context without modifying pattern semantics.

Trading Relationships: `fd:Spread` captures price differentials between contracts, inheriting temporal indirection behavior from constituent tickers.

Specialized Properties: `fd:rollsFrom` and `fd:rollsTo` extend the generic `:changes` property to express contract succession explicitly.

These extensions demonstrate that domain complexity can be added without compromising the core temporal indirection mechanism. The modular architecture ensures that domain-specific extensions can be added without modifying the core pattern abstractions, preserving both pattern integrity and domain expressiveness.

Section 4 validates this approach through implementations in multiple domains, demonstrating how the same instantiation methodology yields robust temporal indirection models regardless of the application context.

4. Validation and Use Cases

This section validates the Temporal Indirection Pattern through implementation and testing in two domains: financial derivatives and air transportation. We demonstrate query execution, measure competency question coverage, and analyze cross-domain applicability in a dedicated GitHub project at https://github.com/YuliaS/temporal_indirection.git.

4.1. Validation Methodology

Our validation approach comprises:

1. **Reference Implementation**: Open-source repository [6] with pattern definition, domain ontologies, and test data
2. **Competency Questions**: SPARQL queries for all competency questions in each domain
3. **Data Coverage**: Three-month temporal spans of instance data
    - All datasets used in this validation were synthetically generated to avoid reliance on proprietary or copyrighted materials. We specified initial value ranges (e.g., plausible price intervals for futures contracts, typical aircraft capacities, and operational time windows) and provided descriptive metadata to guide generation.
    - Instance data was then produced automatically by applying domain-inspired rules and randomized sampling within these ranges. Specific identifiers were fabricated to resemble real-world formats while remaining entirely artificial. This approach ensured that the validation environment reflected realistic conditions while maintaining full compliance with copyright and licensing requirements.
4. **Integrity Validation**: Automated checks for all competency questions, especially gaps, overlaps, and consistency

The companion repository provides a pure-Python reference implementation that loads the pattern ontologies and synthetic datasets, executes the SPARQL competency-question suite, and performs integrity checks using small test harnesses over an in-memory RDFLib graph. Experiments were run with CPython 3.11 (Linux, x86_64) and RDFLib 7.0.0; SPARQL evaluation uses RDFLib's built-in 1.1 engine. Dependencies are limited to RDFLib plus optional utilities (SPARQLWrapper, pandas, tabulate). Reproduction consists of creating a virtual environment, installing `requirements.txt`, and running the domain test scripts. Semantically, we model temporal intervals as half-open [start, end), so "current referent" resolution selects the unique binding with $start \leq t < end$—operationalizing the mapping *m(t)* in Figure 1, aligning with the structural constraints in Figure 2.3, and verified by the competency questions in Tables 3 and 4.

4.2. Financial Derivatives Validation

Table 3 shows how generic competency questions specialize for derivatives trading.

Generic CQ	**Financial Specialization**
CQ1: Current referent	Which contract does CL1 currently reference?
CQ2: Historical resolution	Which contract did CL1 reference on 2024-01-15?
CQ3: Transition timing	When do monthly contract rolls occur?
CQ4: Attribution	What prices are recorded for contract CLG25?
CQ5: Complete history	Show CL1's contract sequence for Q1 2024
CQ6: Rule provenance	Which exchange rules govern CL rolls?
CQ7: Temporal integrity	Verify no gaps in CL1 contract mappings
CQ8: Context reconstruction	Reconstruct spread relationships for date X

Table 3
Financial derivatives competency questions

Key SPARQL queries demonstrate pattern capabilities.
Current Referent Resolution (CQ1):

```
SELECT ?contract WHERE {
  ex:CL1 :hasTemporalBinding ?binding .
  ?binding :bindsTo ?contract ;
           :hasValidTime/:hasStartTime ?start ;
           :hasValidTime/:hasEndTime ?end .
  FILTER(NOW() >= ?start && NOW() < ?end)
}
```

Transition Tracking (CQ3):

```
SELECT ?timestamp ?from ?to ?rule WHERE {
  ?roll a fd:ContractRoll ;
        :hasTimestamp ?timestamp ;
        fd:rollsFrom ?from ;
        fd:rollsTo ?to ;
        :triggeredBy ?rule .
} ORDER BY ?timestamp
```

4.3. Air Transportation Validation

Table 4 shows how generic competency questions specialize for the air transportation use case.

Generic CQ	**Air Transportation Specialization**
CQ1: Current referent	Which aircraft/crew operates UA001 today?
CQ2: Historical resolution	Which aircraft operated UA001 on date X?
CQ3: Transition timing	When do daily aircraft assignments change?
CQ4: Observation attribution	What metrics exist for aircraft N12345?
CQ5: Complete history	What is UA001's aircraft rotation history?
CQ6: Rule provenance	Which scheduling rules govern assignments?
CQ7: Temporal integrity	Do flight assignments have scheduling conflicts?
CQ8: Context reconstruction	Can incident context be fully reconstructed?
CQ9: Maintenance tracking	What maintenance history exists for assigned aircraft?
CQ10: Crew patterns	How do crew rotations align with flights?

Table 4
Specialization of competency questions for air transportation.

The aviation domain adds safety-critical requirements beyond generic competency questions:

- **CQ9**: Track maintenance history for aircraft assigned to flight numbers
- **CQ10**: Analyze crew rotation patterns across flights

These domain-specific extensions validate the pattern's extensibility without modifying core structures.

Key SPARQL queries illustrate validation in this domain.

Context reconstruction (CQ8):

```
SELECT ?aircraft ?captain ?passengers ?severity
WHERE {
        ?incident a avo:SafetyIncident ;
        avo:reportedFor ?operation ;
        avo:hasSeverity ?severity .
        ?operation avo:hasAircraft ?aircraft ;
        avo:hasCrew/avo:hasMember ?captain .
        ?captain avo:hasRole ex:CaptainRole .
OPTIONAL { ?metric :observedReferentEntity ?operation ;
        avo:hasPassengerCount ?passengers } }
```

Maintenance tracking (CQ9):

```
SELECT DISTINCT ?aircraft ?maintType ?maintTime
WHERE {
        ex:UA001 :hasTemporalBinding/:bindsTo ?operation .
        ?operation avo:hasAircraft ?aircraft .
        ?maint avo:involvesAircraft ?aircraft ;
        avo:hasMaintenanceType ?maintType ;
        :hasObservationTime ?maintTime .
        } ORDER BY ?aircraft ?maintTime
```

4.4. Results Summary

Testing across both domains achieved:

- **100% Coverage**: All competency questions answered correctly
- **Temporal Integrity**: No gaps or overlaps detected in the transitions
- **Extensibility**: Domain-specific questions handled without pattern modification

These results confirm the pattern's effectiveness for temporal indirection across heterogeneous domains.

5. Related Work

Having validated our pattern, we now position it relative to existing temporal modeling approaches to clarify our contribution.

5.1. Temporal Qualification vs. Temporal Indirection

Existing temporal approaches can be grouped into four main categories, each revealing the same limitation: they assume identifier-referent relationships remain static within temporal boundaries. Time-indexed relations built on OWL-Time [2] and the Time-indexed Value in Context pattern [7] excel at qualifying when relationships hold but cannot update identifier semantics over time. Situation-based patterns from DOLCE [8] and event models [9] contextualize occurrences but focus on modeling events themselves, rather than the evolution of what an identifier denotes. Fluents and 4D approaches [10] represent entities as temporally extended, emphasizing property changes, but do not address identifier continuity with changing referents. Recent enhancements including RDF-star capabilities in RDF 1.2 [11] enable fine-grained annotation of triples, including temporal metadata, but stop short of providing higher-level patterns for rule-driven reference evolution. In all cases, temporal validity is represented, but the semantics of identifier–referent mapping are assumed to be fixed during any interval, leaving the indirection phenomenon unaddressed.

5.2. Recurring Events vs. Persistent References

The Recurrent Situation Series pattern [12] addresses collections of recurring situations with shared properties. While this pattern successfully models periodicity and series membership, it differs from temporal indirection in key respects: 1) recurrent situations are discrete occurrences rather than continuous reference evolution; 2) each situation maintains its own identity, rather than being accessed through a persistent identifier, 3) the pattern focuses on collection membership, not identifier resolution. Work on calendar-based temporal patterns [13] and periodic events [14] provides mechanisms for expressing complex recurrence rules. These approaches inform our modeling of roll conventions but don't address the identifier-referent indirection layer.

5.3. Domain Ontologies (FIBO) Limitations

The Financial Industry Business Ontology (FIBO)[3, 4], despite containing 2,300+ classes with dedicated derivatives and market data domains, lacks explicit handling of rolling contract mechanisms. While FIBO does include a ReassignableIdentifier class that appears relevant to rolling contracts, its axiomatization fundamentally misaligns with the temporal indirection requirements.

The reassignable identifier is defined as "an identifier that uniquely identifies something for a given time period, and that may be reused to identify something else at a different point in time," with properties `hasAssignmentTerminationDate` and `hasInitialAssignmentDate` that apply directly to the identifier itself. This models scenarios where identifiers are recycled after periods of non-use (like vanity license plates transferred between vehicles), creating temporal gaps between assignments. However, rolling generic tickers like "CL1" exhibit fundamentally different semantics: the identifier maintains continuous existence without assignment/termination dates, while the referent contracts transition seamlessly according to exchange rules. The reassignable identifier's cardinality constraints (max 1 for both assignment dates) enforce discrete reuse periods rather than continuous rule-based transitions.

Furthermore, the class lacks mechanisms to specify transition rules, maintain historical binding sequences, or support the dual attribution required for market observations. The Temporal Indirection Pattern addresses these limitations by separating the persistent identifier (which has no temporal bounds) from temporal bindings (which manage the time-dependent referent relationships), enabling the continuous, rule-governed referent evolution that characterizes financial rolling contracts.

Our pattern fills this gap by explicitly separating identifier persistence from referent dynamics while maintaining complete temporal traceability.

6. Conclusion and Future Work

6.1. Summary of Contributions

This paper presented the Temporal Indirection Ontology Design Pattern, addressing a previously unrecognized gap in temporal knowledge representation. Our contributions include:

- **Conceptual Framework**: Formal separation of persistent identifiers from evolving referents through temporal bindings
- **Reusable Pattern**: Domain-independent abstractions validated across finance and aviation
- **Competency Questions**: Support for current referent resolution, historical reconstruction, and audit trails

- **Open Implementation**: Reference ontologies and validation suites for community adoption

The pattern's success in modeling both financial derivatives and safety-critical aviation operations demonstrates its broad cross-domain applicability.

6.2. Implications

Temporal indirection appears wherever systems require stable references to evolving entities. Beyond our validated domains, potential applications include:

- **Healthcare**: Clinical roles, on-call assignments, equipment allocation
- **Government**: Organizational positions, regulatory interpretations, policy versions
- **Technology**: API versions, service endpoints, deployment configurations

6.3. Future Directions

Future work will extend the foundation by (a) developing a full axiomatization in Description Logic with temporal semantics, supported by reasoning complexity analysis and integration with temporal logic frameworks, (b) conducting performance and scalability benchmarks with large datasets, and (c) exploring template-based instantiation with Reasonable Ontology Templates (OTTR) [15] to streamline adoption by domain experts. We also plan to explore comparative testing of triplestore implementations, characterizing query behavior and identifying optimizations that can support more efficient resolution in temporal indirection scenarios.

In the financial domain, we plan to engage with the Fintech Open Source Foundation (FINOS) [16] to align the pattern with ongoing standards, such as the Common Domain Model (CDM) [17]. In addition, we will seek to support and validate a broader range of financial use cases to further demonstrate the applicability and robustness of the pattern.

Establishing this pattern could benefit ontology engineering efforts across finance, government, healthcare, and other domains, enabling consistent modeling of roles, identifiers, and evolving entities. In conclusion, the Temporal Indirection Ontology Design Pattern provides a principled solution to representing temporal indirection in ontologies—a previously missing element in the ontology design patterns toolkit.

Declaration on Generative AI

During the preparation of this work, the author used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] T. Lebo, S. Sahoo, D. McGuinness, PROV-O: The PROV Ontology, W3C Recommendation (2013). URL: http://www.w3.org/TR/2013/REC-prov-o-20130430/.

[2] S. Cox, C. Little, Time Ontology in OWL, W3C Candidate Recommendation Draft (2022). URL: https://www.w3.org/TR/owl-time/.

[3] M. Bennett, The Financial Industry Business Ontology: Best practice for big data, Journal of Banking Regulation 14 (2013) 255–268.

[4] Enterprise Data Management Council, Financial industry business ontology (FIBO), https://github.com/edmcouncil/fibo, 2025. Accessed: 2025-09-22.

[5] V. Presutti, A. Gangemi, Dolce+ d&s ultralite and its main ontology design patterns, in: Ontology Engineering with Ontology Design Patterns, IOS Press, 2016, pp. 81–103.

[6] Y. Svetashova, Temporal indirection pattern: An ontology design pattern, 2025. URL: https://github.com/YuliaS/temporal_indirection. doi:10.5281/zenodo.16914828.

[7] S. Peroni, D. Shotton, F. Vitali, Scholarly publishing and linked data: describing roles, statuses, temporal and contextual extents, in: Proceedings of the 8th International Conference on Semantic Systems, 2012, pp. 9–16.

[8] S. Borgo, R. Ferrario, A. Gangemi, N. Guarino, C. Masolo, D. Porello, E. M. Sanfilippo, L. Vieu, DOLCE: A descriptive ontology for linguistic and cognitive engineering, Applied ontology 17 (2022) 45–69.

[9] W. R. Van Hage, V. Malaisé, R. Segers, L. Hollink, G. Schreiber, Design and use of the simple event model (SEM), Journal of Web Semantics 9 (2011) 128–136.

[10] C. Welty, R. Fikes, S. Makarios, A reusable ontology for fluents in OWL, in: FOIS, volume 150, 2006, pp. 226–236.

[11] O. Hartig, P.-A. Champin, G. Kellogg, A. Seaborne, RDF 1.2 concepts and abstract syntax, W3C Working Draft (2025). URL: https://www.w3.org/TR/rdf12-concepts/.

[12] V. A. Carriero, A. Gangemi, A. G. Nuzzolese, V. Presutti, An ontology design pattern for representing recurrent situations, in: Advances in pattern-based ontology engineering, IOS Press, 2021, pp. 166–182.

[13] H. J. Ohlbach, Calendrical calculations with time partitionings and fuzzy time intervals, in: International Workshop on Principles and Practice of Semantic Web Reasoning, Springer, 2004, pp. 118–133.

[14] A. Tuzhilin, J. Clifford, On periodicity in temporal databases, Information Systems 20 (1995) 619–639.

[15] M. G. Skjæveland, D. P. Lupp, L. H. Karlsen, J. W. Klüwer, Ottr: Formal templates for pattern-based ontology engineering, in: Advances in Pattern-Based Ontology Engineering, IOS Press, 2021, pp. 349–377.

[16] FINOS, The Fintech Open Source Foundation, About FINOS, https://www.finos.org/about-us, 2025. Accessed: 2025-09-22.

[17] FINOS, The Fintech Open Source Foundation, Common Domain Model (CDM), https://github.com/finos/common-domain-model, 2025. Accessed: 2025-09-22.