Transforming semi-structured life science diagrams into meaningful domain ontologies with DiDOn

C. Maria Keet

School of Computer Science, University of KwaZulu-Natal, Private Bag X54001, 4000 Durban, South Africa, tel: +27 31 260 1035, fax: +27 31 260 7001, email: keet@ukzn.ac.za

Abstract

Bio-ontology development is a resource-consuming task despite the many open source ontologies available for reuse. Various strategies and tools for bottom-up ontology development have been proposed from a computing angle, yet the most obvious one from a domain expert perspective is unexplored: the abundant diagrams in the sciences. To speed up and simplify bio-ontology development, we propose a detailed, micro-level, procedure, DiDOn, to formalise such semi-structured biological diagrams availing also of a foundational ontology for more precise and interoperable subject domain semantics. The approach is illustrated using Pathway Studio as case study.

Keywords: Ontology Development, OWL, Bio-Ontology, Diagram, Pathway, BFO

1. Introduction

Ontologies can be used as an important mechanism to link and integrate data, databases, and conceptual data models, thanks to providing a logic-based representation of the domain of interest that is independent of specific applications. Linking and integrating can be done through annotation of instances across databases with a domain ontology, such as the widely used Gene Ontology [1] and similar OBO ontologies [2], linking ontologies to conceptual data models [3, 4] or linking an ontology directly to data in different sources by means of a mapping layer [5, 6]. Increasingly, an ontology is also seen as an end in itself, whereby it is deployed as a way to represent the knowledge of a particular subject domain [7, 8] and may be used for hypothesis elimination by reducing the theoretical options to those that are logically consistent with the formally represented theory before commencement of laboratory experiments [9, 10], and scientific discovery [11]. Also, with ontologies, one can avoid duplication of costly research and manage the exponentially growing amount of data to push science forward [1, 12]. However, development of ontologies is a resource-intensive task where non-ontological resources—'legacy' representations of the scientific knowledge—are, often manually, consulted to ensure adequate coverage and to ease the ontology development process. The statistical analysis of Simperl et al's [13] survey of 148 ontology development projects showed that "domain analysis was shown to have the highest impact on the total effort" of ontology development and the "par-

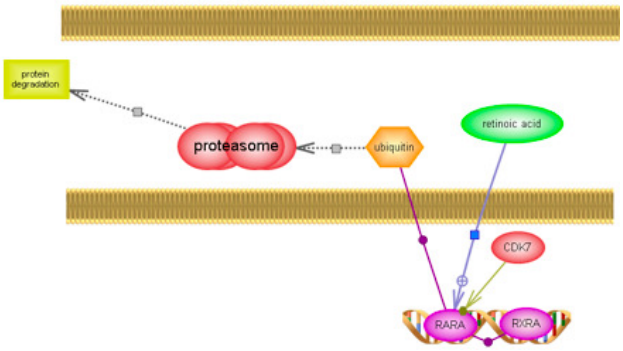

Figure 1: A diagram representing the interactions between the various molecules in a pathway: degradation of the RAR and RXR by the proteasome.

ticipants shared the view that process guidelines tailored for [specialised domains such as health care, or in projects relying on end-user contributions] are essential for the success of ontology engineering projects". Some efforts have gone into automating this bottom-up development process, ranging from database reverse engineering [14] to natural language processing (NLP) [15, 16], to ontologising thesauri [17]. The former two use relatively generic algorithms and heuristics and are therefore noisy so that they require adaptation for bio-ontology learning to yield useable results [18], whereas the latter is still more manual than automated. Invariably, the approaches show that the procedure to go from an informal non-ontological resource to a logic-based ontology is too large to carry out in a single step.

An alternative, and hitherto unexplored, option for bottom-up ontology development is to analyse what can be done with the many *diagrams* in biology that contain icons and stylised drawings for biological entities. This could be especially useful since the life sciences are very diagram-oriented and there are plenty of scientific drawing tools. It is surprising that such semi-structured diagrams, like the one depicted in Figure 1, have not been used to find (candidate) classes and relationships for domain ontologies beyond the initial exploration by [7] for the ISEE drawing tool and the SMBL Harvester tool under development [19] that is geared toward ontologising SMBL-annotated diagrams for *in silico* simulations. In addition, if they are used, they can provide the sought-after *intermediate representation* that domain experts are familiar with and computer scientists also can handle [20, 21]. Once formalised in a comprehensive ontology, consistency of those biological theories can be checked with automated reasoners and the opportunity enhanced for scientific discovery.

The aim is to fill this hiatus in bottom-up ontology development. This requires two principle components in order to lay solid foundations: a formalisation step and adequate treatment of the subject domain semantics. In this paper, we propose a procedure to formalise such semi-structured biological diagrams, i.e., from *Di*agram to *D*omain *On*tology, DiDOn, focussing on how to formalize it whilst being faithful to the subject domain semantics. The formalisation aims at two common usage scenarios: (*i*) the so-called

low-hanging fruit with OBO or SKOS and its use for data annotation and computational linguistics, and (*ii*) to capture the necessary details for theory analysis by formalising it in a very expressive (Semantic Web) ontology language, with OWL 2 DL as a minimum. Both require a *micro-level* method to represent the formal and ontological details of the diagram vocabulary in an expressive ontology beyond just classes and some of their relationships so as to include guidance also for the axioms and ontological quality criteria, which subsequently functions as a seed ontology to automatically formalise the diagrams themselves by means of a transformation algorithm. In turn, this can be integrated into *macro-level* methodologies that provide the high-level process-oriented information systems perspective for ontology development. The DiDOn micro-level formalization procedure will be demonstrated using the biochemical pathway modelling tool Pathway Studio (PS) [22, 23] as use case. Its vocabulary is analysed and then categorised with a foundational ontology so that the icons are given both a formal semantics and a precise subject domain semantics.

The remainder of the paper is organised as follows. Section 2 considers the choice of a representation language and modelling choices following the selection of a foundational ontology, which feed into the DiDOn procedure in Section 2.4. We apply the procedure to the Pathway Studio graphical modelling tool in Section 3. Section 4 contains a discussion and Section 5 the conclusions.

2. Methods: How to Formalise it?

After a brief assessment of the principal differences in proposed approaches to and methodologies for bottom-up ontology development, we shall look at the issue of choosing a suitable language and the consideration to use a foundational and/or other ontologies. The outcome is summarised as the DiDOn procedure in Section 2.4.

2.1. Extant Methods and Methodologies

There are comprehensive methodologies for ontology development, such as METHONTOLOGY [20], MOKI [21], the NeON methodology [24], OntoSpec [25], and the "Ontology Development 101" (OD101) [26]. The former three mention non-ontological resource reuse, but they do not elaborate on how exactly this is to be carried out, other than an NLP approach and indicating reuse of thesauri. For instance, one such recent application is NLP for pharmacogenomics ontology development, which requires manual rule construction and 'normalization' of verbs (candidate relationships) into relationships using the PHARE domain ontology and finally is represented in a semantic network [16]. Although PHARE is an OWL ontology, it uses only the $\mathcal{ALCHIF}(D)$-fragment, and it does not use any foundational ontology, such as GFO [27], RO [28], BFO [29], DOLCE [30], or SUMO [31], to enforce precision on the subject domain semantics and to make it interoperable with other bio-ontologies. Concerning independent methods for reusing 'legacy' representations that can be incorporated into the macro-level methodologies, then there is one study similar

in spirit to the one proposed here [7], which is, however, with a simple graphical modelling tool containing only 4 core elements, and therefore still relies on much manual analysis, and it does not provide a structured, reusable, approach toward formalisation. The OBO Foundry [2] has a set of resources and principles [8], but a method is yet to be developed that goes beyond manual examination of scientific literature to extract content that has to be represented in the bio-ontology.

OntoSpec focuses on formalising subject domain knowledge in detail, uses the DOLCE-OS language and is informed by DOLCE and OntoClean, but it does not include non-ontological resource usage. OD101 predates inclusion of expressive and standardised ontology languages and ontology reuse and interoperation. Nevertheless, the interesting aspect of these two approaches is that, to the best of our knowledge, they are the only methodologies with formalisation guidelines detailing *how* to go from informal to logic-based representations with instructions how to include the axioms and which ones are better than others, i.e., the micro-development compared to the macro-development steps of the other methodologies that revolve around designing and deploying, say, a waterfall methodology, that includes instructions how to plan for development of an ontology and maintain it, but not how to formalise it.

The 'how to formalise it?' question by itself is not new neither in IT and Computing [32, 33] nor in logic, be that going from natural language to first order logic or its interaction with the graphical classical "blocks world" [34]. Regarding formalising it *in an ontology* as opposed to a mere logical theory, OntoClean and the so-called "ontological level" can be added [35, 36], which provide reasons why one formalization is better than another. Their commonalities lie in the considerations for choosing a logic language to formalise it, the ontological commitments, and the realisation that they do affect the representation of the subject domain, hence, also any procedure to formalise the diagrams.

2.2. Formalization in Different Languages

We briefly outline which languages typically are, or can be, used for bio-ontology development, which differ in expressiveness and encoding peculiarities, and introduce informally the first steps of formalizing the diagrams.

2.2.1. The Language

Regarding formalisation of the icons in the biological diagrams, the first aspect is to choose a suitable logic-based language. This depends on the scope and purpose of the ontology (if there is one at all) and the desired reasoning services (if any) [37]. Current usage of bio-ontologies fall broadly into two categories: annotation of resources, such as data in databases and text in scientific literature with the Gene Ontology [1] and similar ontologies, and scientific ontologies representing the knowledge of a subject domain, such as the Foundational Model of (human) Anatomy [38] and the BioPax ontology for biological pathways [39]. The former group of ontologies require support for navigation, queries to retrieve a simple class in the hierarchy,

and scalability at the class-level and at the instance-level; hence, a language with low expressiveness suffices for querying and is required for scalability, such as the Open Biological and biomedical Ontologies' `obo`-format (a directed acyclic graph), the W3C standardised Simple Knowledge Organisation System (SKOS) language (essentially RDF) [40], and the W3C standardised OWL 2 EL and OWL 2 QL profiles [41]. A scientific ontology requires a very expressive language to represent fine distinctions between the entities and reasoning services such as satisfiability of the ontology, classification of classes in the hierarchy, and complex class queries. One can choose any language, be it full first order predicate logic (FOL), an extension (e.g., temporal), or a standardised decidable fragment of FOL to guarantee termination of the reasoning services and foster interoperability and reuse with other ontologies. The second option indicates that the most expressive language OWL 2 DL [42] may be suitable. This information is summarised in Figure 2, which can be extended with more ontology languages.

Expressiveness of the Representation Language. Of the languages mentioned, only FOL has the expressiveness to represent n-ary relations, with $n \geq 3$. Reification of an n-ary to n binary relations requires identification constraints among those n binaries for it to be semantically equivalent to the original n-ary, but none of the other above-mentioned languages has this language feature, hence, n-aries can only be approximated. Note though, that not all perceived n-aries are real n-aries; i.e., some can be split into binaries without losing information, whereas others cannot. This modelling issue is well-known in relational database theory and conceptual data modelling as assessment of *functional dependencies* and methods exist to disentangle it [32, 43]. For instance, the ternary about HIV transmission, $\forall x, y, z(transmission(x, y, z) \rightarrow HIVsubtype(x) \wedge Donor(y) \wedge Recipient(z))$ ("HIVsubtype transmission from Donor to Recipient"), is not further decomposable without losing information, whereas the ternary $\forall x, y, z(works(x, y, z) \rightarrow Doctor(x) \wedge Department(y) \wedge Building(z))$ ("Doctor works for Department in Building") *can* safely be split into two binary relationships without losing information: $\forall x, y(works\_for(x, y) \rightarrow Doctor(x) \wedge Department(y))$ and $\forall x, y(works\_in(x, y) \rightarrow Doctor(x) \wedge Building(y))$.

One can foray into extensions of FOL, Description Logics (DL) and OWL, so that one can represent temporal, fuzzy, probabilistic, or rough knowledge. For instance, one may insist upfront that one has to be able represent that, say, Hepatitis *normally* has fever as symptom [44] and multiple similar cases, which can be dealt with using probabilistic default knowledge: let $(\phi, \psi)[l, u]$ stand for "generally, if an object belongs to ϕ, then it belongs to ψ with a probability in [l,u]" in a probabilistic extension of OWL [45], then $(\exists \texttt{hasSymptom.Fever} \mid \texttt{Hepatitis})[1, 1]$. Another common request is to be able to represent ordered sequences of entities or events, such as the chemical reactions in a metabolic pathway, that may require a temporal logic to represent and reason adequately over such knowledge. However, these extensions have not made it to mainstream ontology development yet.

There are more aspects one may want to consider, such as the fine-grainedness of a language (e.g.,

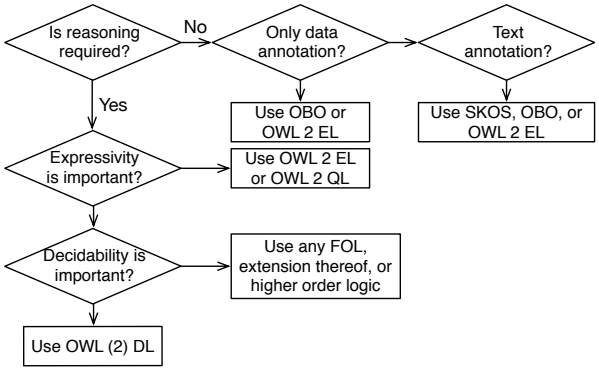

Figure 2: Decision diagram to choose a suitable language, indicating current typical usage and suggestions for use.

if one can represent not only relationships, but also the components of a relationship) and its semantics (e.g., graph-based, model-theoretic), which is interesting from a logic and philosophical perspective, but less relevant for practical ontology development and therefore not pursued here.

Encoding Peculiarities. Domain ontology developers tend to distinguish between what it is 'understood to represent' and the computational representation. That is, one can take, say, Cell to be a universal, class, or concept, and represent it as an OWL class `Cell` in an ontology, but one also can store Cell in a database table, by which it mathematically has become an instance yet 'think of it' and pretend it to be a universal, class, or concept. This is of particular relevance for SKOS and some OBO ontologies. For instance, the Gene Ontology is downloadable in OBO or OWL format—i.e., its taxonomy consists of, mathematically, classes—and as a database—i.e., mathematically it is a taxonomy of instances. This need not concern the subject domain experts, but it does affect how the ontology can be used in ontology-driven information systems. For SKOS, there is no such choice: each particular SKOS 'concept' is serialised as an instance, regardless whether one models it directly into SKOS or transforms an OWL ontology into SKOS. For instance, if one would have chosen to represent that insuline is a subtype of a peptide, then one declares the SKOS `ex:peptide rdf:type skos:Concept` and `ex:insulin ex:broaderGeneric ex:peptide` which are, in the mathematical sense, about instances. Hence, also in an OWL to SKOS transformation, each OWL class becomes a SKOS instance due to the mapping of `skos:Concept` to `owl:Class` [46]. This is a design decision of SKOS, that, once known, can be handled easily in the application layer, in a similar way to GO.

A different encoding peculiarity is to exploit OWL 2 punning features to squeeze second-order rules in a first-order setting to avail of its software infrastructure: convert all class-level classes and expressions (the TBox) into individual assertions (ABox), encode the second-order rules in the TBox and classify the classes-converted-into-individuals accordingly. An example of this approach is the application of OntoClean

[47].

Hence, one has to be careful with the distinction between the 'intended meaning' and the actual encoding.

2.2.2. First Steps Toward Formalization of Biology Diagrams

As some point in the bio-ontology development, one has to choose a representation language. For both categories of scenarios mentioned in the introduction, the first step of the formalization is to assess the "icon vocabulary" of the diagram drawing tool for unary 'object-like' entities and n-ary ($n \geq 2$) 'relationship-like' entities. Among the n-aries, one then distinguishes between generic relationships, such as parthood, participation, dependence, and constitution, and other recurring relationships, which are the general relationships specific to the domain, such as development, regulation, and transformation in the life sciences subject domain. From here onwards, the formalisation steps differ for the chosen languages. The relatively straight-forward procedure for OBO and SKOS is included in the guidelines in Section 2.4 only.

For OWL 2 DL, one also assess a few sample diagrams to check for cardinality restrictions, i.e., if a particular (type of) n-ary is linked to more than one unary, and checks for sequences of the same or different n-aries, which indicate possible transitivity or property chaining. n-aries where $n \geq 3$ can be approximated by reification without the identification constraint, but this makes the overall ontology logically complicated and difficult to understand for the domain expert, and therefore should be used sparingly even if one were to use an n-ary ontology design pattern.

For an arbitrary expressive logic language, there are more options to consider, such as spatiality and temporality, which both feature in many diagrams implicitly. Spatiality is often represented with sections of different background colour, lipid bi-layers, or the name of the (type of) cell, tissue, or organ, therefore requiring inclusion of both spatial relations as well as spatial entities at the appropriate level of granularity. Temporal aspects are normally represented as chains of unaries and n-aries with indicative labels like transports, transcribes, or flows, though, in general, the temporal dimension has not been investigated widely for ontologies. For both spatial and temporal extensions, a wide range of theories are available (we will return to this in Section 4). n-aries where $n \geq 3$ can be represented as such.

2.3. Foundational Ontology Modelling Choices

Most real bio-ontologies do not exist in isolation, but are linked to other ontologies, be they other domain ontologies or foundational ontologies. Using a foundational ontology with its generic categories of entity types and core relationships across subject domains can facilitate bio-ontology interoperation [2], it speeds up ontology development, and it has been shown to improve its quality [48]. Some of such ontologies are DOLCE [30], BFO(+RO) [29, 28], GFO [27], and SUMO [31].

The principal modelling choice they introduce, is that it forces one to choose between n-aries as unaries (classes in OWL) or n-aries as n-aries (object properties in OWL). An intuitive formalization of the n-aries

is to keep them as such, so that there is a close correspondence with the original diagram; this easily can be done also in OWL and any arbitrary FOL language. Foundational ontologies, however, have a separate branch for 'processes' (Perdurant in DOLCE and Occurrent in BFO) and relate this with a *new* relation to 'objects' (Endurant in DOLCE, Continuant in BFO), such that an endurant is a participant in a perdurant. For instance, the person Mary is a participant in a running instance that, in turn, is part of a marathon, but not that there is a 1-to-1 formalisation for "Mary runs a marathon" where "runs" is the label for a binary relation between Mary and the marathon she is running. Thus, a biological diagram icon may be an arrow denoting regulation, which can be formalised as a binary relationship `regulates` or `regulatedBy`, or as an unary predicate (OWL class) `Regulation` as subtype of DOLCE's Process or BFO's ProcessualEntity. The former results in a more compact representation, is intuitively closer to the domain expert's understanding, and makes it easier to verbalise the ontology, and therefore is likely to be more useful in praxis. The latter is more generic, and thereby likely to increase reusability of the ontology. At the time of writing, it has not been determined experimentally which option is better for domain ontologies.

In addition, *dependency* or *inherence* has to be addressed, which has the meaning that a depends on b if and only if, necessarily, b is present whenever a is present. One can represent the role or function, `a`, an entity plays as a subtype of its bearer `b`, such as $\forall x(Student(x) \rightarrow Person(x))$, but also—and in foundational ontologies in particular—such that one creates a hierarchy for the roles and one for the bearers and relate the entities through a *dependency* relationship. The dependent entities are represented as subclasses of NonPhysicalEndurant in DOLCE (DependentContinuant in BFO) and their bearers as PhysicalEndurant in DOLCE (IndependentContinuant in BFO). Clearly, the latter is a more elaborate encoding, but perhaps more interoperable than the former.

A minor issue is the representation of an attribute—a binary relationship between a class and a data type, like OWL's data property—because, from the viewpoints of foundational ontologies and interoperability, they ought not to be represented as such in an ontology. Put differently, inclusion of a subject domain-specific data property denotes an application decision, therewith decreasing the ontology's value to solve application integration and interoperability problems, and it does not capture 'what it is' ontologically. For instance, both hasColour $\mapsto$ Rose $\times$ `String` and hasColour $\mapsto$ Rose $\times$ `Integer` refer to the very same property Colour, not two intrinsically different 'colour-things'. Instead, the foundational ontologies' approach is to reify such attributes to a unary predicate (OWL class, BFO universal etc.) and add them as subtypes of, e.g., BFO's Dependent continuant or Quality without the values of the attributes, or, in a similar fashion, to add them as subtypes of DOLCE's Quality and use Quale for the value regions (as approximation of data type). In this way, the semantic agreement between attributes can be asserted through the vocabulary in the ontology. In a diagram language like Pathway Studio, this modelling choice does not arise, but drawing tools such as STELLA/ISEE do have icons representing certain amounts of matter or mixture, like Water, in a "stock" icon that may have some "converter" icon Pollutant concentration [7], which makes it rather inviting to simply

add it as a data property in an OWL ontology instead of the DOLCE/BFO approach with qualities as unaries.

At the time of writing, there are no full mappings between the extant foundational ontologies, so one has to choose one; how to choose the most appropriate one for the task at hand and why that one, is a separate task, which is beyond the current scope.

2.4. DiDOn Procedure to Formalise a Diagrammatic Vocabulary

The considerations and decision points described in the previous sections are structured into the *Di*agram to *D*omain *On*tology, DiDOn. This has a preliminary step of requirements analysis (item 0; more detail can be found elsewhere), then the core steps with the analysis of the elements in the graphical language and how to represent them formally in the ontology (items 1-6), from which the rules to populate the ontology with entities from the individual diagrams can be constructed (item 7). If desired, steps 1-6 can be enforced in a software-mediated formalisation workflow and/or incorporated into existing macro-level methodologies for ontology development. The procedure is as follows.

0. The usual ontology requirements analysis (such as scope, purpose, sample usage, type of queries, desired reasoning services, etc.), including
   (a) Choose a representation language, informed by, among others, Figure 2 and Table 1 in [37];
1. Basic assessment of the icons in the tool's "legend":
   (a) Divide between unaries, binaries ($n = 2$), and n-aries where $n \geq 3$;
   (b) For the n-aries where $n \geq 3$: assess the functional dependencies and create new relationships with lower arity, where possible (use a procedure described in, e.g., [32, 43]);
   (c) Divide the binaries and real n-aries by generic relationships (like parthood, participation), domain-specific relationships, and attributes;
2. Use OBO? If no: go to Item 3; If yes, do:
   (a) Represent each unary as an OBO `Class` (node in the graph);
   (b) Add parthood as `part-of` that is transitive;
   (c) Binaries, choose:
      i. As relations: Add the domain-specific binaries as user-defined OBO `Relation` (edge in the graph), and omit the attributes and n-ary relations ($n \geq 3$);
      ii. As classes: Represent each binary as an OBO `Class` (node in the graph);
   (d) If the result of item 2a is multiple hierarchies or ontologies and the result of item 2c includes user-defined relations (e.g., `inheres_in`), then, optionally:
      i. Use the BFO-in-`obo` to clarify the hierarchies;
      ii. Use the RO-in-`obo` for compatibility of the relations with other OBO-ontologies;

(e) Add so-called cross-products [49] within or across these and, optionally, other domain ontologies, where applicable;

(f) Proceed to item 8;

3. SKOS? If no: go to Item 4; If yes, do:

   (a) Declare unaries to be a SKOS entity, using
   `ex:unary_name rdf:type skos:Concept`,
   where `ex` is shorthand of the declared namespace URI and `unary_name` should be replaced with the unary;

   (b) Extend SKOS with a limited notion of the 'subsumption' relation, using
   `ex:broaderGeneric rdfs:subPropertyOf skos:broader`
   replacing `ex` with the appropriate URI;

   (c) If parthood exists among the binaries, extend SKOS with
   `ex:broaderPartitive rdfs:subPropertyOf skos:broader`,
   replacing `ex` with the appropriate URI;

   (d) Add the other (non-attribute) binaries as `skos:related` or extend the SKOS RDF Schema accordingly (analogous to the extensions in items 3b and 3c);

   (e) Proceed to item 8;

4. Choose a foundational ontology, or a module thereof.

5. N-aries as classes? If no: go to Item 6; If yes, do:

   (a) Declare unaries as unary predicates subsumed by `Continuant` (or `Endurant` in DOLCE);

   (b) Declare *part-of*, *participates-in*, and *depends-on* (or: *inheres-in*) as binary relationships, if not already present from the chosen foundational ontology, and type the relationships:

      i. Add `part-of` and/or `has-part` and its relational properties (to the extent possible in the language), and assess if proper parthood is needed as well;

      ii. Add `participates-in` and/or `has-participant` and declare its domain as, as a minimum, `Continuant` and range as `Occurrent` (or `Perdurant` in DOLCE);

      iii. Add `inheres-in` (or: `depends-on`) and declare its domain as `DependentContinuant` (or `NonPhysicalEndurant`) and range as `IndependentContinuant` (or `PhysicalEndurant`);

   (c) Declare all domain-specific n-aries as classes, suitably positioned as a subtype of `Occurrent`;

   (d) Add the attributes as unaries subsumed by `Quality` and, if available in the foundational ontology:

      i. Add the abstract representation of the data types under `Quale`, spatial/temporal `Region`, or similar;

      ii. Add the relationships between the class, `Quality` and `Quale` (in DOLCE, they are `qt` and `ql`, respectively);

   (e) Consider also sample diagrams;

i. An n-ary has relations to > 1 unary? If yes: record that cardinality will have to be included when processing the diagrams;

(f) Proceed to item 8;

6. N-aries as relationships. Do:

(a) Consider also sample diagrams:

i. An n-ary has relations to > 1 unary? If yes: note cardinality;

ii. Chaining of n-aries? If yes: note concatenation;

iii. Sequences of the same binary? If yes: declare transitivity;

(b) Use OWL 2 DL? If no: go to Item 6c; If yes, do:

i. Declare unaries as subclasses of (a suitable subclass of) `Continuant` in the class hierarchy;

ii. Include (or ascertain inclusion when using `owl:import` to import the foundational ontology) generic object properties, as described in item 5b, but for the *part-of* object property characteristics, declare it only transitive and reflexive;

iii. Declare domain-specific binaries as OWL object properties;

iv. Attributes: choose either

A. Under `Quality`, like in item 5d; or

B. As OWL data property and declare a suitable domain (an OWL class) and range (XML data type);

v. n-aries with $n > 2$? If yes: if used often, drop it, if used sparingly, do the approximation of reification;

vi. Proceed to item 8;

(c) Full FOL or more. Do:

i. Examine at least the spatial and temporal dimension;

A. If spatial relations, then consider inclusion of an RCC or mereotopological theory;

B. If temporal relations, then consider inclusion of the Allen temporal relations and consider formalisation in a temporal logic;

ii. Are there any "system" icons? If yes: consider granularity;

iii. Declare binaries and n-aries;

iv. Declare same as in items 5a, 5b, 5d, and 6a;

v. Proceed to item 8;

7. Consider the contents of relevant domain ontologies, if any, and assess their commitments regarding the choices made, i.e., item 5 vs. item 6, item 6b vs. item 6c, and item 6(b)iv, then do either:

(a) If the same choices have been made:

i. import the ontology in whole or in part, as appropriate;

ii. declare equivalences between the classes and relations of the imported ontology and the

formalised diagram vocabulary;

(b) If different choices have been made: remodel the knowledge in whole or in part to match the choices and include it, as appropriate;

8. Ontology population by processing the diagrams: see below;

$\geq$ 9. Ontology maintenance (verification of represented knowledge, update in the light of recent advances in science, etc.) and deployment.

Item 8 consists of writing an algorithm to process each icon in an individual diagram to the appropriate class, relationships, or axiom(s) in the seed ontology, based on the formalisation pattern obtained in steps 1-6, and, in case another other domain ontology has been imported into the ontology under development, then one should include a subroutine to check if the class, relationship, or axiom is already present. More precise suggestions can be made upfront, especially with respect to OBO and SKOS, but less so for the more expressive formalisations, because they depend to a larger extent on the choices made in steps 4, 5, or 6. The basic idea for generating the rules is similar for each option (with variations in notation), except for the distinction between handling n-aries as classes or as relatioships:

A. SKOS:

(a) Each label of the icon in a diagram, A, generates an assertion

```
ex:A rdf:type skos:Concept
```

and

```
ex:A skos:broaderGeneric ex:unary_name
```

where `ex` is shorthand for the URI, `unary_name` an unary declared previously in item 3a, and `A` the label of the icon in the particular diagram that has the same icon as `unary_name` has.

(b) Add the `ex:broaderPartitive` and `skos:related` assertions between the newly added entities, following the choices made in item 3d;

(c) Proceed to item D;

B. N-aries as classes:

(a) OBO:

i. Each label of the matching icon in a diagram generates a new `ID` and `name` in the ontology and is a new child of (i.e., `is_a:`) the respective main class added in items 2a and 2c;

ii. Perform post-processing with the added entities, depending on the choices made in item 2d;

iii. Proceed to item D;

(b) OWL:

i. Each label of the matching icon in a diagram, A, generates a new OWL class and a

```
SubClassOf(A unary_name)
```

assertion for the corresponding class added in item 5a, 5c, and 5d;

ii. For each `unary_name` added in item 5c, add `has-participant` assertions, such that all its

players ($\texttt{C}_i$, $i \geq 2$, subsumed by `Continu-ant`) in the original n-ary appear in
`ObjectPropertyDomain(has-participant unary_name)`
`ObjectPropertyRange(has-participant` $\texttt{C}_i$`)`
and, depending on the recorded cardinality (item 5(e)i), add the appropriate OWL assertions;

iii. For each `unary_name` added in item 5d, add the respective object property assertions for the quales, if applicable;

iv. Proceed to item D;

(c) Full FOL or more: Depends on the language chosen, and the choices made particularly in items 6(c)i, 6(c)ii, and 6(c)iv;

C. N-aries as relationships:

(a) OBO:

i. Each label of the icon in a diagram generates a new `ID` and `name` in the ontology and is a new child of (i.e., `is_a:`) the respective main class added in item 2a;

ii. Use `relationship:part_of` with part-whole assertions, and
`relationship:[user-defined-relation-in-item-2c]`
for the other relations;

iii. Perform post-processing with the added entities, depending on the choices made in item 2d;

iv. Proceed to item D;

(b) OWL:

i. As in item B(b)i, but applied to the outcome of item 6(b)i;

ii. For each binary in a diagram, with corresponding object property name added in item 6(b)iii, add `ObjectPropertyDomain` and `ObjectPropertyRange` restrictions, and, where applicable, the cardinality restrictions;

iii. Process the attributes in the diagrams according to the choice made in item 6(b)iv;

iv. Proceed to item D;

(c) Full FOL or more: Depends on the language chosen, and the choices made particularly in items 6(c)i, 6(c)ii, and 6(c)iv;

D. Substitute any abbreviation used in the diagrams with its full name.

E. If a unary from formalisation steps 1-6 denotes an individual, then either convert its subclasses into individuals or use a layered architecture for the ontology with a branch for particulars and one for categories.

We shall see an application of a more detailed ontology population algorithm for OWL with n-aries as relationships in the case study.

3. Case Study: Analysis of the Pathway Studio Graphical Vocabulary

The purpose of the case study is to demonstrate the usability and detailed workings of the procedure (as opposed to hiding that in an application). To this end, a graphically fairly comprehensive modelling tool, Pathway Studio (PS) [23], was chosen out of one of many graphical pathway tools [50]. PS lets the user build pathways and analyse them on the relationships between genes, proteins, cell processes and diseases. It is useful for illustrative purpose, because of the expressiveness of its vocabulary and the generation of its source data. The source data for the PS diagrams that come with the installation originate from a combination of NLP of scientific literature and manual examination, correction, and curation. Given the difficulty of ontology learning through NLP tools, one can envision a two-step process from manual examination and NLP of scientific articles to PS, and from PS to expressive ontology, which, in turn, can be used to improve NLP. To give an idea of its usage, and thereby providing hints toward desired reasoning scenarios, hence, contributing to choosing a representation language, several specific user requirements were abstracted from the biological examples: Compound X (e.g., a potential drug) binds and activates Y, which is a main switch in pathway Z that should be interrupted to cure the disease. Sample queries are:

Q1: Is Y involved in some other pathway?

Q2: What are the characteristics of the other pathways that Y is involved in? E.g., are they spatially separated (e.g., in different tissues), is there an analogue in another species?

Q3: Given that one binds and activates Y with X, is there an activation of some Y' by X that is also a signaling molecule in pathway Z'?

Q4: Is it known that there is some endogenous X' similar to X that also binds and activates Y?

Q5: Which As and Bs are affected by Y's X and X' in location C? (abstracted from [51][1])

Q1 can be a simple class-query, but the others show that OBO and SKOS are insufficient to meet the desired inferences, hence OWL or arbitrary FOL should be chosen to formalise the Pathway Studio Vocabulary (PSV). To foster interoperability with other ontologies, OWL 2 DL is chosen. Note that this still permits simplification to an NLP ontology for text mining to find data for the diagrams or for data annotation, as well as an extension to a more expressive language, which will be discussed in Section 4.

3.1. Pathway Studio Vocabulary Inventory

Let us now start with the first step in the DiDOn procedure: the basic analysis of the PSV. The following high-level informal descriptions are intended to give a non-biochemist an indication of the kind of things in the PSV depicted in Figure 3. Subject domain semantics especially useful in the formalization is italicized when they first appear.

[1] "identify proteins and cell processes mediated through androgen receptor signaling using an androgen receptor agonist (17β-trenbolone) and antagonist (flutamide) in the liver."

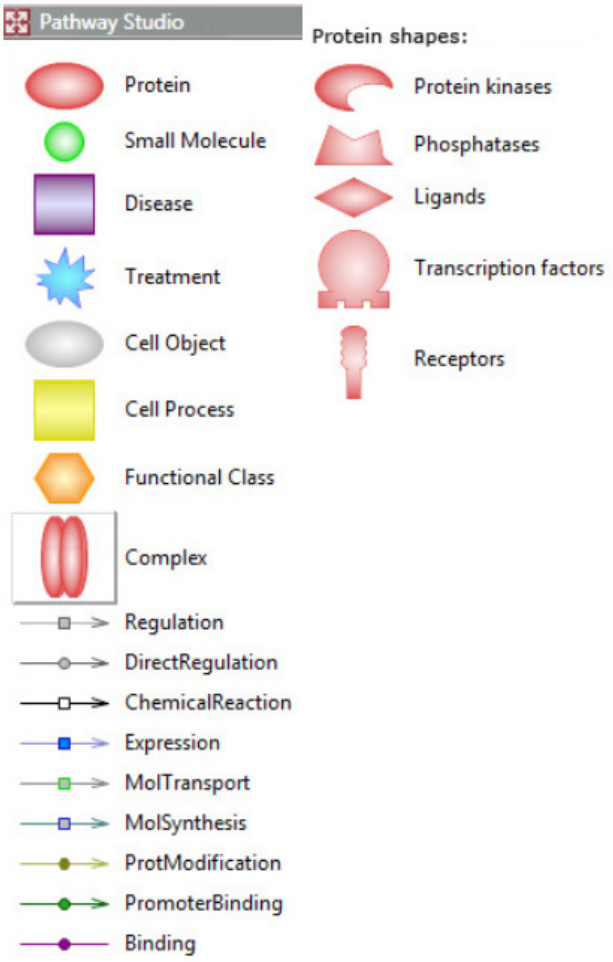

Figure 3: Pathway Studio's icons.

- Protein: biopolymer consisting of many linked α-ami-no acids. Shapes denote subclasses:
  - Kinases are enzymes, i.e., proteins with a *function/role* to catalyse the addition of a phosphate to a molecule (opposite of phosphatases).
  - Phosphatases are also enzymes, catalysing the removal of a phosphate from a molecule.
  - Ligand is a molecule that *binds* to a receptor, sometimes also referred to as substrate;
  - Transcription factor is a molecule that binds to a transcription factor binding site and thereby *regulates* expression of a gene that is *located* relatively nearby the site where that molecule binds.
  - Receptor: the receiving molecule in a receptor-ligand binding, i.e., a molecule with a role; depending on the location, it can be a protein when [*in/on*] a membrane or a Nuclear receptor (purple oval) bound to DNA, hence, a receptor is generally the less-mobile one of the receptor-ligand binding (the ligand 'arrives at' the receptor).

- Small Molecule is a pervasive notion that appeals to intuition for its informal description : refers to, e.g., glucose, nitric oxide, (not 'macromolecule', such as protein, starch); there is no strict cut-off point for the physical size of the molecule.
- Treatment represents a *system*, here: a *cascade* of processes in which molecules *participate*;
- Cell Object represents a combination of a structural entity at a certain *location* in the cell, includes organelles;
- Cell Process represents a (combination of) process(es) located in the cell;
- Functional Class: molecule with a particular function;
- Complex *consists of* at least one protein and at least one other molecule (that also can be a protein) that may be bound to it.
- The arrows are binary *relations*, with Mol an abbreviation of molecule, and Prot of protein modification. A "$\oplus$" in the arrow means *positive* effect and a line *negative* effect of the type of *interaction* indicated by the arrow's colour.

Hence, there are unaries and binaries only, no attributes, and no typical generic relationships in the PSV.

3.2. Foundational Ontology Choice

From the available foundational ontologies, using BFO [29] (see also Figure 4) is a strategic option, because many bio-ontologies align themselves with it, even though all other extant foundational ontologies are more comprehensive.

The main issue to decide is to formalise PS's binaries as classes or as object properties. The limited set of arrows (Figure 3, bottom) suggests formalising them as OWL object properties, and it concurs with the relations of the Relation Ontology (RO) [28] that is integrated with BFO and its extensions under consideration. The dependency relationship in BFO terminology is inheres_in and holds between dependent and independent continuants.

Domain ontologies may be useful for the individual diagrams. Given the named categories (Figure 3, top section), BioPax [39] will be useful to consult, which covers metabolic pathways and molecular interactions, as well as the OWLized Gene Ontology, Cell Cycle Ontology, and Protein Ontology [52]. The standardised categorisations, such as nomenclatures for enzymes [53] and nuclear receptors [54] are yet to be OWLized.

3.3. The Formalisation of the PSV

Having committed to OWL 2 DL, BFO+RO, and n-aries as relationships, we need to assess sample diagrams; a small one is depicted in Figure 1 and many larger ones can be consulted online [22]. This reveals that regulation is transitive, each pathway has at least three molecules, and nuclear receptors are bound to exactly one DNA molecule.

Figure 4: Graphical rendering of a section of the BFO v1.1 taxonomy.

We can now proceed to the formalization in OWL 2 DL. The relevant basics of OWL 2 DL, hence, the *formal* semantics, and its more compact DL notation are included in Appendix A. Together with BFO for the *subject domain* semantics, we can proceed to the actual formalisation of the icons. Due to space limitations as well as the principle of minimum necessary commitment, only the conservative axioms will be described here. Let ***BFO*** be shorthand denoting all its 39 classes {`Entity, Continuant, Occurrent, ...`} (see also Figure 4), hence in its OWLized format $BFO \in V_C$. Let ***RO*** be shorthand notation to denote all 12 relationships {`has_participant, has_part, has_agent, ...`} in the Relation Ontology, hence in its OWLized format $RO \in V_{OP}$. BFO and RO classes and object properties are henceforth indicated with ***italic courier*** font to distinguish them from the classes and object properties generated from the formalisation of the PSV. Combining this with the basic analysis of the PSV, we obtain the following set of basic assertions.

`Protein` $\sqsubseteq$ `Molecule`,

`SmallMolecule` $\sqsubseteq$ `Molecule`,

`ProteinComplex` $\sqsubseteq$ `Molecule`,

`Molecule` $\sqsubseteq$ *`Object`*,

`TranscriptionFactor` $\sqsubseteq \exists$`inheres_in.Protein`,

`Ligand` $\sqsubseteq \exists$`inheres_in.Protein`,

$\texttt{Receptor} \sqsubseteq \exists \texttt{inheres\_in}.\texttt{Protein}$,

$\texttt{Enzyme} \sqsubseteq \exists \texttt{inheres\_in}.\texttt{Protein}$,

$\texttt{FunctionalClass} \sqsubseteq \texttt{Function}$,

$\texttt{FunctionalClass} \sqsubseteq \exists \texttt{inheres\_in}.\texttt{Molecule}$,

$\texttt{Kinase} \sqsubseteq \texttt{Enzyme}$, $\texttt{Phosphatase} \sqsubseteq \texttt{Enzyme}$,

$\texttt{Kinase} \sqsubseteq \neg \texttt{Phosphatase}$,

$\texttt{CellProcess} \doteq \mathit{Process} \sqcap \exists \mathit{located\_in}.\texttt{Cell}$,

$\texttt{CellObject} \doteq \mathit{Object} \sqcap \exists \mathit{contained\_in}.\texttt{Cell}$,

$\texttt{ExtracellularProtein} \doteq \texttt{Protein} \sqcap$
$\exists \mathit{located\_in}.\neg \texttt{Cell}$,

$\texttt{NuclearReceptor} \doteq \texttt{Receptor} \sqcap =1\ \texttt{binds}^{-}.\texttt{DNA}$,

$\texttt{ProteinComplex} \doteq \texttt{Complex} \sqcap$
$\exists \mathit{has\_part}.\texttt{Protein} \sqcap \exists \mathit{has\_part}.\texttt{Molecule} \sqcap$
$\forall \mathit{has\_part}.(\texttt{Protein} \sqcup \texttt{Molecule})$,

$\exists \texttt{binds}.\texttt{TranscriptionFactorBindingSite} \sqsubseteq$
$\texttt{TranscriptionFactor}$,

$\texttt{Pathway} \sqsubseteq \texttt{System} \sqcap \geq 3\ \mathit{has\_part}.\texttt{Molecule} \sqcap$
$\forall \mathit{has\_part}.\texttt{Molecule}$,

$\texttt{Treatment} \sqsubseteq \exists \mathit{has\_participant}.\texttt{Molecule}$,

$\texttt{System} \sqsubseteq \mathit{GenericallyDependentContinuant}$,

$\texttt{binds\_promoter} \sqsubseteq \texttt{binds}$,

$\texttt{binds} \sqsubseteq \texttt{reacts\_chemically}$,

$\texttt{up\_regulates} \sqsubseteq \texttt{regulates}$,

$\texttt{regulates\_directly} \sqsubseteq \texttt{regulates}$,

$\texttt{modifies} \sqsubseteq \texttt{reacts\_chemically}$,

$\texttt{modifies\_protein} \sqsubseteq \texttt{modifies}$,

$\exists \texttt{modifies\_protein}^{-} \sqsubseteq \texttt{Protein}$,

$\texttt{synthesis} \sqsubseteq \texttt{reacts\_chemically}$,

$\texttt{molecular\_synthesis} \sqsubseteq \texttt{synthesis}$,

$\texttt{molecular\_transport} \sqsubseteq \texttt{transports}$,

$\exists \texttt{molecular\_transport}^{-} \sqsubseteq \texttt{Molecule}$.

The declared subclasses of `Molecule` are disjoint.

`Receptor`, `TranscriptionFactor`, `Enzyme`, and `Ligand` are disjoint and subclasses of *Role*.

`regulates`, `expresses`, `transports`, and `reacts_chemically` are sub-properties of `topObjectProperty`.

Practically, both BFO and RO were imported into the new seed ontology with an `owl:import` statement and the above-listed statements were added to create the combined ontology `OWLPathS.owl`, which is available online at `http://www.meteck.org/files/ontologies/OWLPathS.owl`. Although BFO can be represented in the simple $\mathcal{ALC}$ DL language, the DL characterisation of OWLPathS is $\mathcal{SHIQ}$, i.e., indeed requiring OWL 2 DL expressivity.

Finally, we have to assess the seed ontology obtained so far with related domain ontologies. BioPax has chosen to use n-aries as classes, not relations, hence, aside from the straightforward class equivalences such as pathway, complex, protein, and small molecule, it will require remodelling efforts, such as matching BioPax' class `BiochemicalReaction` with the `owlpaths:reacts_chemically` object property and handle the conversion between `participates_in` and domain and range axiom. The Protein Ontology already fits exactly with the choices made here. Equivalences have to be asserted between, among others, `owlpaths:Protein` and `PR:000000001` (PRO's protein) and PRO's protein complex (in turn, an imported `GO:0043234`) and `owlpaths:ProteinComplex`.

3.4. Ontology Population with the Diagrams

Given this core formalization of the PSV, a set of rules has to be be devised to automate the ontology learning process. Partial algorithms for the polygons and arrows are included in Algorithm 1 and 2, respectively. To illustrate its design and workings, let us consider Figure 1 again: the label ubiquitin is associated with a FunctionalClass-icon, hence an assertion

$\texttt{Ubiquitin} \sqsubseteq \texttt{FunctionalClass}$

should be added to the ontology. `Ubiquitin` is a (specifically) dependent continuant (because `FunctionalClass` is), so it has to inhere in some molecule; in this case, this is `Protein`, motivating the addition of

$\texttt{Ubiquitin} \sqsubseteq \exists \texttt{inheres\_in.Protein}$

to the ontology. The purple protein-shape has a label RARA, which is an abbreviation of Retinoic Acid Receptor Alpha, i.e., it is a nuclear receptor, thus one can add

$\texttt{RetinoicAcidReceptorAlpha} \sqsubseteq \texttt{NuclearReceptor}$

and it also inheres in a protein, and so forth for the other elements. This second step with `inheres_in` applies only when the protein shape is not a red oval.

For cell processes, like the yellow rectangle labeled protein degradation, one first adds

$\texttt{ProteinDegradation} \sqsubseteq \texttt{CellProcess}$

so that it inherits from `CellProcess` that it is `located_in` the `Cell`, and, optionally, refines the range either to a subclass or a part of `Cell` later on. The process is similar for all polygons, such that the class is added under one of the main categories described in the previous section. In addition, it checks for any equivalences with PRO to avoid duplication. The algorithm is also extended with the BFO+RO-based BioTop [55], to avoid duplication and enhance the precision of the axioms; e.g., while Ubiquitin is not yet in BioTop and thus

can be added, multiple cell parts are included already, so that one can retrieve that partonomy and select the appropriate location (e.g., `Cytoplasm`).

The algorithm for the relationships is determined by the colour, adornments, and direction of the arrow, and that the assertion for the direction has to be added in the inverse. The reason for the latter is that while retinoic acid *expresses* RARA (see Figure 1), this is not the case for *all* retinoic acid molecules; conversely, it does hold that RARA is *expressed by some* retinoic acid molecule, hence

$\texttt{RARA} \sqsubseteq \exists \texttt{expressed\_by.RetinoicAcid}$

is the appropriate axiom to add to the ontology. This works analogously with the other relationships. A partial design-level algorithm for the arrows is described in Algorithm 2, which can be executed after Algorithm 1.

This completes one execution path through the DiDOn procedure.

4. Discussion

As the PS case study demonstrates, one clearly can obtain a lot of information from the diagrams for an expressive bio-ontology. The DiDOn procedure in Section 2.4 aids in this process. One may ask why a formalisation procedure and why specifically the DiDOn procedure is any good. Concerning the former, a procedure brings the formalization decisions to the fore, requires the developer to make the modelling decisions explicit, and be able to coherently communicate and document that so that within as well as across ontology development projects this can be harmonised, or at least made clear. Concerning the latter, first, it has to be noted that there are no extant AI techniques that transform the bio-diagrams into bio-ontologies in a structured way, although the SMBL to OWL transformation under way with SMBL Harvester may have a promising intersection with the work presented here. Second, it goes beyond one-off community of practice by expounding a method of formalization of the informal 'legacy' resources to standardised knowledge representation languages that is sufficiently generic to work with any graphical language of bio-diagrams, yet not too generic to render it of little use for biological resources. Moreover, third, it incorporates foundational ontology use to also handle subject domain semantics as opposed to a mere formalization into an arbitrary logical theory. Consequences of this aspect are the inclusion of the hitherto neglected representation decision to represent n-aries as classes vs n-aries as relationships, and the reuse of classes and relationships that are also used in other bio-ontologies and adjusting the representation of attributes so as to foster interoperability upfront, which improve the quality of the ontology as has been demonstrated in, e.g., [48]. Fourth, it acknowledges that different migration paths may be viable, and how, ensuring all essential tasks are carried out in a consistent manner and therewith making repeatability of the

process possible and opening up the way for structured transformations[2] and linking of ontologies that took a different formalisation route and therewith facilitating ontology interoperability. The OWLPathS seed ontology with its formalisation in OWL 2 DL, import of BFO+RO foundational ontologies and link with BioTop and PRO meets all these quality features. Given the relative comprehensiveness of the procedure, DiDOn may contribute to standardising decisions made for and during the formalisation within and across domain ontology development projects or even be incorporated in the drawing tools.

The procedure, however, does not help with the transformation of some of the implicit information that requires subject domain knowledge, such as knowing that a kinase is an enzyme. Other bottom-up approaches face this hurdle to an even greater extent: NLP for science has to use both general NLP rules and bio-adjusted heuristics [15], (e.g., the suffix "-ase" denotes the name of an enzyme).

4.1. OBO and SKOS

Having demonstrated DiDOn with OWL+BFO+relationships leaves room for remarks on transformations to OBO and SKOS. It is useful to note that if one already has one formalisation, then the others can be obtained at least semi-automatically. There are online scripts that transform an OWL file into OBO or SKOS; an example of `OWLPathS` converted to `SKOSPathS` with Manchester's OWLtoSKOS converter [56] is available online at `http://www.meteck.org/files/ontologies/SKSOPathS`. As described in Section 2.2.1, classes become instances in SKOS and each usage of a unary in the actual diagrams is `skos:broaderGeneric` its respective core entity.

In the direction from SKOS to OWL, and provided one wants a real ontology and not some arbitrary OWL file, then once again one has to choose a foundational ontology, decide to represent n-aries as relations or as classes, and choose a suitable expressive language, i.e., following steps 4-7 in DiDOn. Some work to at least partially automate this idea has been carried out [57].

Concerning item 4 of the procedure—the use of a foundational ontology—one may question why it was not placed before OBO and SKOS. No foundational ontology is available in SKOS or OBO, other than an OBO version of BFO. Aside from BFO-in-`obo`, one also could choose to develop a BFO-in-SKOS and add that to step 3, and/or extend the SKOS vocabulary with the Relation Ontology relations. All other extant foundational ontologies require a much more expressive language to enable representation of the intended semantics (BFO too, in fact, as we will discuss in Section 4.2). Nevertheless, one could argue, even an 'ultra-ultra-light' version may be useful. However, this also makes it easier to introduce errors in the transformation due to the lower precision and accuracy that later on has to be re-analysed to represent it correctly anyway. A hitherto unexplored alternative option might be to use the expressive ontology for

[2]For instance, between n-ary-as-class and n-ary-as-relationship: switch between a predurant and its two `participates_in` axioms and an object property with its domain and a range axiom.

modelling only and to remove axioms as required by the application scenario, instead of removing them upfront.

4.2. Extensions

Given that OWL 2 DL was chosen as ontology language and not an arbitrary FOL language, time and location have not been addressed in the formalization, because they are even more deeply embedded in the diagrams and, at the time of writing, no practically usable technological solution for temporal ontologies exist yet that lets one use it with automated reasoning. Time is implicit with the very notion of pathway—i.e., some specific sequence of interactions—that is approximated with the arrows. Efforts to try to capture this with just "precedes" and "immediately precedes" relationships in an a-temporal ontology language bears no formal semantics, hence cannot be used in automated reasoning. Neither OBO nor OWL is expressive enough to assert that "a immediately precedes b" means that we have not only $(\mathtt{a}, t)$ and $(\mathtt{b}, t')$ but also that it must hold for the time points that $\neg\exists t''.t < t'' < t'$. This has as consequence that the automated reasoner cannot infer anything about those assertions and cannot detect inconsistencies among assertions about time. In addition, it pulls the lid off the temporal knowledge representation and reasoning box, which also contains the Allen temporal relations (such as during, overlap, and disjoint) and the Time Ontology [58], among others. The Time Ontology, as well as 'precedes' labels, do provide a-temporal OWL object properties that can suffice when one only wants to *annotate* resources with a time component. Few temporal DLs exist, of which $\mathcal{DLR}_{\mathcal{US}}$ [59] is very expressive and could be used for ontology-as-scientific-theory to represent the knowledge and TDL-Lite [60] is so-called computationally well-behaved, but seriously limited in expressiveness.

Regarding location, consider Figure 1's two thick lines representing membranes: because there are two and the bottom part of the figure shows a DNA helix, one can infer that the second line represents the membrane of the nucleus (hence, an eukaryotic cell) and the first one the cell membrane. Generally, compartmentalization is represented with such lines, different shaded areas, or both; examples from other diagram software can be examined for, e.g., IUBMB's intracellular pathways [61]. Inferring the implicit location is not easy due to both how it is represented in the diagrams—geometrical shapes but also, say, a stylised ribosome or heart—and (mereo)topological representation and reasoning is not solved for decidable languages and scalability of terminological and instance reasoning [62, 63]. In addition, recollect sample question Q2 in Section 3, which in natural language text moves easily between the molecule-level of the pathway in a cell and its nucleus to its location in some tissue, thus indicating the need to take into account granularity of representation and cater for cross-granular queries, which does not have a clear counterpart in the diagrams. What may be feasible to handle are the Cell Process icons, such as Protein Degradation (Figure 1) that involves several reactions, hence is a common 'folding' operation [64]. One also may want to modularize the knowledge along those lines, using an arbitrary-logic or OWL-based technique [65, 66], or use OWL

`owl:import` statements to link the pathways.

All these topics are active fields of research.

Nevertheless these potentially missing aspects of the diagrams—potentially, because they are not necessary for all ontologies and ontology-driven information systems and do not occur in all diagrammatic languages—the basic formalization of the icon vocabulary already provides a solid basis to simplify and speed up ontology development compared to manual efforts or NLP. In addition, using a more expressive language invites the domain expert to be more precise so as to resolve ambiguities, a benefit which was already observed in [7] for eco-ontologies. Only then can it be checked computationally if the many diagrammatic pathways are consistent together and gaps can be found easily, which motivates further modelling in case the missing knowledge was known or can serve as impetus for laboratory experimentation. Added benefits of the approach are that such diagrams also can be deployed as intermediate representation of the knowledge so as to facilitate understanding and communication between logicians and the content providers. Also, it can bring the information modelled in such diagrams—often hidden or locked in, e.g., expensive hardcopy textbooks—into the open access domain for free use and reuse.

We are in the process of taking DiDOn to the implementation-level. This comprises, on the one hand, zooming in on converting diagrams to an ontology in the subject domain of microbiology and health with, initially, opportunistic infectious diseases, and, on the other hand, developing full software-support for DiDOn (a proof-of-concept tool for Step 4 of DiDOn is already available [67]).

5. Conclusions

To speed up and simplify bio-ontology development, we proposed the DiDOn procedure to formalise semi-structured life science diagrams, which operates at the micro-level of ontology development by providing a structured approach to representing different pieces of information in the ontology. DiDOn is aimed at extracting explicit and implicit knowledge from diagram-based 'legacy' resources in such a way so that also the subject domain semantics can be preserved and that it can be carried out in a clear, traceable, and reproducible way. Four trajectories for formalization were identified—OBO, SKOS, OWL 2 DL, and arbitrary FOL—with the option to integrate it with a foundational ontology so that both a formal and precise subject domain semantics is generated when populating the ontology. This was demonstrated with the extensive icon vocabulary of Pathway Studio and OWL 2 DL, BFO, and n-aries as binary relations.

Acknowledgment

The author gratefully acknowledges Kristina Hettne for providing several models of NHR pathways and sample queries.

Appendix A. The OWL 2 DL Ontology Language

The OWL 2 DL direct semantics (a model-theoretic semantics) can be consulted online [42] and the essentials of DLs are described in [68]. Here we summarise the OWL 2 DL syntax and semantics only insofar as is necessary to have a self-contained paper; the interested reader is referred to [42] for further details. By standard OWL notation, OP denotes an object property, OPE denotes an object property expression, C denotes a class, and CE a class expression.

Definition 1 (OWL 2 DL Ontology Syntax (abbreviated))**.** A vocabulary $V = (V_C, V_{OP}, V_I)$ over a datatype map D (as formalised in [42]) is a 3-tuple consisting of the following elements:

- V_C is a set of classes;
- V_{OP} is a set of object properties;
- V_I is a set of individuals;

Definition 2 (OWL 2 DL Ontology Semantics (abbreviated))**.** Given a datatype map D and a vocabulary V over D, an interpretation $I = (\Delta_I, \cdot^C, \cdot^{OP}, \cdot^I)$ for D and V is a 4-tuple with the following structure:

- Δ_I is a nonempty set called the object domain;
- $\cdot^C$ is the class interpretation function that assigns to each class $C \in V_C$ a subset $(C)^C \subseteq \Delta_I$;
  $\cdot^C$ is extended to class expressions as follows
  - `ObjectAllValuesFrom(OPE CE)`, $\{x \mid \forall y : (x,y) \in (OPE)^{OP}$ implies $y \in (CE)^C\}$;
  - `ObjectSomeValuesFrom(OPE CE)`, $\{x \mid \exists y : (x,y) \in (OPE)^{OP}$ and $y \in (CE)^C\}$;
  - `ObjectMinCardinality(n OPE CE)`, $\{x \mid \sharp\{y \mid (x,y) \in (OPE)^{OP}$ and $y \in (CE)^C\} \geq n\}$;
  - `ObjectExactCardinality(n OPE CE)`, $\{x \mid \sharp\{y \mid (x,y) \in (OPE)^{OP}$ and $y \in (CE)^C\} = n\}$;
  - `ObjectComplementOf(CE)`, $\Delta_I \setminus (CE)^C$;
- $\cdot^{OP}$ is the object property interpretation function that assigns to each object property $OP \in V_{OP}$ a subset $(OP)^{OP} \subseteq \Delta_I \times \Delta_I$ and such that $\cdot^{OP}$ is extended to `Inv(OP)` with the meaning $\{(x,y) \mid (y,x) \in (OP)^{OP}\}$;
- $\cdot^I$ is the individual interpretation function that assigns to each individual $a \in V_I$ an element $(a)^I \in \Delta_I$.

Further, with respect to satisfaction of OWL 2 DL class expression axioms in interpretation I w.r.t. ontology O, the class axioms `SubClassOf(CE`$_1$ `CE`$_2$`)` holds if $(CE_1)^C \subseteq (CE_2)^C$, `EquivalentClasses(CE`$_1$ `...CE`$_n$`)` if $(CE_j)^C = (CE_k)^C$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$, `DisjointClasses(CE`$_1$ `...CE`$_n$`)` if $(CE_j)^C \cap (CE_k)^C = \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$. Regarding property axioms, `SubObjectPropertyOf(OPE`$_1$ `OPE`$_2$`)` if $(OPE_1)^{OP} \subseteq (OPE_1)^{OP}$, and for the relevant object property characteristics only transitivity: `Trans(OPE)`, $\forall x, y, z : (x,y) \in (OPE)^{OP}$ and $(y,z) \in (OPE)^{OP}$ implies $(x,z) \in (OPE)^{OP}$ (parthood is reflexive, antisymmetric, and transitive, but antisymmetry cannot be expressed in OWL 2 DL and reflexivity only on simple object properties).

Given that OWL 2 DL is based on Description Logics (DL), we shall use the more concise DL notation. For instance, the DL statement

$\texttt{Protein} \sqsubseteq \texttt{Molecule}$

i.e., all individuals that are proteins are also molecules, can be represented equivalently in FOL as

$\forall x(Protein(x) \rightarrow Molecule(x))$

and in OWL 2 DL functional syntax as

`SubClassOf(Protein Molecule).`

The `ObjectSomeValuesFrom` in Definition 2 is the serialised rendering of the existential quantification ($\exists$), an `Inv(OP)` is denoted as $\texttt{OP}^-$, `ObjectComplementOf(C)` as $\neg\texttt{C}$, and `EquivalentClasses(`$\texttt{CE}_1$ $\texttt{CE}_2$`)` as $\texttt{CE}_1 \doteq \texttt{CE}_2$.

References

[1] Gene Ontology Consortium . Gene Ontology: tool for the unification of biology. Nature Genetics 2000;25:25–9.

[2] Smith B, Ashburner M, Rosse C, Bard J, Bug W, Ceusters W, et al. The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 2007;25(11):1251–5.

[3] El-Ghalayini H, Odeh M, McClatchey R, Arnold D. Deriving conceptual data models from domain ontologies for bioinformatics. In: 2nd Conference on Information and Communication Technologies (ICTTA'06). IEEE Computer Society; 2006, p. 3562 –7.

[4] Sugumaran V, Storey VC. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. ACM Transactions on Database Systems 2006;31(3):1064–94.

[5] Calvanese D, Keet CM, Nutt W, Rodríguez-Muro M, Stefanoni G. Web-based graphical querying of databases through an ontology: the WONDER system. In: Proceedings of ACM Symposium on Applied Computing (ACM SAC'10). ACM; 2010, p. 1389–96. March 22-26 2010, Sierre, Switzerland.

[6] Poggi A, Lembo D, Calvanese D, De Giacomo G, Lenzerini M, Rosati R. Linking data to ontologies. J on Data Semantics 2008;X:133–73.

[7] Keet CM. Factors affecting ontology development in ecology. In: Ludäscher B, Raschid L, editors. Data Integration in the Life Sciences 2005 (DILS2005); vol. 3615 of *LNBI*. Springer Verlag; 2005, p. 46–62. San Diego, USA, 20-22 July 2005.

[8] Smith B, Ceusters W. Ontological realism: A methodology for coordinated evolution of scientific ontologies. Applied Ontology 2010;5:79–108.

[9] Bandini S, Mosca A. Mereological knowledge representation for the chemical formulation. In: 2nd Workshop on Formal Ontologies Meets Industry 2006 (FOMI2006). Trento, Italy; 2006, p. 55–69.

[10] Keet CM, Roos M, Marshall MS. A survey of requirements for automated reasoning services for bio-ontologies in OWL. In: Proceedings of the 3rd Workshop on OWL: Experiences and Directions (OWLED 2007); vol. 258 of *CEUR-WS*. 2007,6-7 June 2007, Innsbruck, Austria.

[11] Wolstencroft K, Stevens R, Haarslev V. Applying OWL reasoning to genomic data. In: Baker C, Cheung H, editors. Semantic Web: revolutionizing knowledge discovery in the life sciences. Springer: New York; 2007, p. 225–48.

[12] Madin JS, Bowers S, Schildhauer MP, Jones MB. Advancing ecological research with ontologies. Trends in Ecology & Evolution 2008;23(3):159–68.

[13] Simperl E, Mochol M, Bürger T. Achieving maturity: the state of practice in ontology engineering in 2009. International Journal of Computer Science and Applications 2010;7(1):45–65.

[14] Lubyte L, Tessaris S. Automated extraction of ontologies wrapping relational data sources. In: Proc. of DEXA'09. 2009,.

[15] Alexopoulou D, Wächter T, Pickersgill L, Eyre C, Schroeder M. Terminologies for text-mining; an experiment in the lipoprotein metabolism domain. BMC Bioinformatics 2008;9(Suppl 4).

[16] Coulet A, Shah NH, Garten Y, Musen M, Altman RB. Using text to build semantic networks for pharmacogenomics. Journal of Biomedical Informatics 2010;43(6):1009–19.

[17] Soergel D, Lauser B, Liang A, Fisseha F, Keizer J, Katz S. Reengineering thesauri for new applications: the AGROVOC example. Journal of Digital Information 2004;4(4). URL `http://journals.tdl.org/jodi/article/view/jodi-126/111`.

[18] Liu K, Hogan WR, Crowley RS. Natural language processing methods and systems for biomedical ontology learning. Journal of Biomedical Informatics 2011;44(1):163–79.

[19] SMBL harvester. Online; 2011 (last accessed: July 2011). URL `http://code.google.com/p/sbmlharvester/`.

[20] Fernandez M, Gomez-Perez A, Pazos A, Pazos J. Building a chemical ontology using METHONTOLOGY and the ontology design environment. IEEE Expert: Special Issue on Uses of Ontologies 1999;January/February:37–46.

[21] Ghidini C, Kump B, Lindstaedt S, Mabhub N, Pammer V, Rospocher M, et al. Moki: The enterprise modelling wiki. In: Proceedings of the 6th Annual European Semantic Web Conference (ESWC2009). 2009,Heraklion, Greece, 2009 (demo).

[22] Ariadne Genomics . Pathway studio. Online; Last accessed: Aug 30, 2011. URL `http://www.ariadnegenomics.com/products/pathway-studio/`.

[23] Nikitin A, Egorov S, Daraselia N, Mazo I. Pathway studio–the analysis and navigation of molecular networks. Bioinformatics 2003;19(16):2155–7.

[24] Suarez-Figueroa MC, de Cea GA, Buil C, Dellschaft K, Fernandez-Lopez M, Garcia A, et al. NeOn methodology for building contextualized ontology networks. NeOn Deliverable D5.4.1; NeOn Project; 2008.

[25] Kassel G. Integration of the DOLCE top-level ontology into the OntoSpec methodology. Technical Report HAL : hal-00012203/arXiv : cs.AI/0510050; Laboratoire de Recherche en Informatique d'Amiens (LaRIA); 2005. Http://hal.archives-ouvertes.fr/ccsd-00012203.

[26] Noy N, McGuinness D. Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, and Stanford Medical Informatics Technical Report SMI-2001-0880; Stanford Knowledge Systems Laboratory; 2001.

[27] Herre H, Heller B. Semantic foundations of medical information systems based on top-level ontologies. Knowledge-Based Systems 2006;19:107–15.

[28] Smith B, Ceusters W, Klagges B, Köhler J, Kumar A, Lomax J, et al. Relations in biomedical ontologies. Genome Biology 2005;6.

[29] BFO . Basic Formal Ontology. (last accessed August 2011). URL `http://www.ifomis.org/bfo`.

[30] Masolo C, Borgo S, Gangemi A, Guarino N, Oltramari A. Ontology library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003).; 2003. Http://wonderweb.semanticweb.org.

[31] Niles I, Pease A. Towards a standard upper ontology. In: Welty C, Smith B, editors. Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001). 2001,Ogunquit, Maine, October 17-19, 2001.

[32] Halpin T. Information Modeling and Relational Databases. San Francisco: Morgan Kaufmann Publishers; 2001.

[33] ter Hofstede AHM, Proper HA. How to formalize it? formalization principles for information systems development methods. Information and Software Technology 1998;40(10):519–40.

[34] Barwise J, Etchemendy J. The language of first-order logic. Stanford, USA: CSLI Lecture Notes; 3rd ed.; 1993.

[35] Guarino N, Welty C. An overview of OntoClean. In: Staab S, Studer R, editors. Handbook on ontologies. Springer Verlag; 2004, p. 151–9.

[36] Guarino N. The ontological level: Revisiting 30 years of knowledge representation. In: Borgida A, et al., editors. Mylopoulos Festschrift; vol. 5600 of *LNCS*. Springer; 2009, p. 52–67.

[37] Keet CM. Dependencies between ontology design parameters. International Journal of Metadata, Semantics and Ontologies

2010;5(4):265–84.

[38] Rosse C, Mejino Jr JLV. A reference ontology for biomedical informatics: the foundational model of anatomy. J of Biomedical Informatics 2003;36(6):478–500.

[39] Demir E, Cary MP, Paley S, Fukuda K, Lemer C, Vastrik I, et al. The BioPAX community standard for pathway data sharing. Nature Biotechnology 2010;28(9):935–42.

[40] Miles A, Bechhofer S. SKOS Simple Knowledge Organization System Reference. W3C Recommendation; World Wide Web Consortium (W3C); 2009. URL `http://www.w3.org/TR/skos-reference/`.

[41] Motik B, Grau BC, Horrocks I, Wu Z, Fokoue A, Lutz C. OWL 2 Web Ontology Language Profiles. W3C Recommendation; W3C; 2009. URL `http://www.w3.org/TR/owl2-profiles/`.

[42] Motik B, Patel-Schneider PF, Grau BC. OWL 2 web ontology language: Direct semantics. W3C Recommendation; W3C; 2009. URL `http://www.w3.org/TR/owl2-direct-semantics/`.

[43] Abiteboul S, Hull R, Vianu V. Foundations of databases. Addison Wesley, USA; 1995.

[44] Schulz S, Stenzhorn H, Boekers M, Smith B. Strengths and limitations of formal ontologies in the biomedical domain. Electronic Journal of Communication, Information and Innovation in Health (Special Issue on Ontologies, Semantic Web and Health) 2009;3(1):31–45.

[45] Lukasiewicz T, Straccia U. Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics 2008;6(4):291–308.

[46] Isaac A, Summers E. SKOS Simple Knowledge Organization System Primer. W3C Standard; World Wide Web Consortium; 2009. Http://www.w3.org/TR/skos-primer.

[47] Glimm B, Rudolph S, Völker J. Integrated metamodeling and diagnosis in OWL 2. In: Patel-Schneider PF, Pan Y, Hitzler P, Mika P, Zhang L, Pan JZ, et al., editors. Proceedings of the 9th International Semantic Web Conference; vol. 6496 of *LNCS*. Springer; 2010, p. 257–72.

[48] Keet CM. The use of foundational ontologies in ontology development: an empirical assessment. In: Antoniou G, et al., editors. 8th Extended Semantic Web Conference (ESWC'11); vol. 6643 of *LNCS*. Springer; 2011, p. 321–35. Heraklion, Crete, Greece, 29 May-2 June, 2011.

[49] Mungall CJ, Bada M, Berardini TZ, Deegan J, Ireland A, Harris MA, et al. Cross-product extensions of the Gene Ontology. Journal of Biomedical Informatics 2010;44(1):80–6.

[50] Suderman M, Hallett M. Tools for visually exploring biological networks. Bioinformatics 2007;23(20):2651–9.

[51] Martyniuk CJ, Alvarez S, McClung S, Villeneuve DL, Ankley GT, Denslow ND. Quantitative proteomic profiles of androgen receptor signaling in the liver of fathead minnows (*pimephales promelas*). Journal of Proteome Research 2009;8(5):2186–200.

[52] Natale DA, et al. The Protein Ontology: a structured representation of protein forms and complexes. Nucleic Acids Research 2011;39(Database issue):D539–45.

[53] Enzyme Nomenclature. 1992. URL `http://www.chem.qmul.ac.uk/iubmb/enzyme/`.

[54] Nuclear Receptors Committee . A unified nomenclature system for the nuclear receptor subfamily. Cell 1999;97:1–20.

[55] Beisswanger E, Schulz S, Stenzhorn H, Hahn U. BioTop: An upper domain ontology for the life sciences - a description of its current structure, contents, and interfaces to OBO ontologies. Applied Ontology 2008;3(4):205–12.

[56] OWLtoSKOS converter. Last accessed: April 2011. URL `http://owl.cs.manchester.ac.uk/owltoskos/`.

[57] Hepp M. SKOS to OWL. Online; Last accessed: Aug 30, 2011. URL `http://www.heppnetz.de/projects/skos2owl/`.

[58] Hobbs JR, Pan F. An ontology of time for the semantic web. ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing 2004;3(1):66–85.

[59] Artale A, Franconi E, Wolter F, Zakharyaschev M. A temporal description logic for reasoning about conceptual schemas and queries. In: Flesca S, Greco S, Leone N, Ianni G, editors. Proceedings of the 8th Joint European Conference on Logics

in Artificial Intelligence (JELIA-02); vol. 2424 of *LNAI*. Springer Verlag; 2002, p. 98–110.

[60] Artale A, Kontchakov R, Lutz C, Wolter F, Zakharyaschev M. Temporalising tractable description logic. In: Proc. of the 14th International Symposium on Temporal Representation and Reasoning (TIME-07). 2007,Alicante, June 2007.

[61] Sigma-Aldrich metabolic Pathways Map. Online; Last accessed: August 31, 2011. URL `http://www.sigmaaldrich.com/life-science/metabolomics/learning-center/metabolic-pathways.html`.

[62] Wessel M. Obstacles on the way to qualitative spatial reasoning with description logics: some undecidability results. In: Goble CA, McGuinness DL, Möller R, Patel-Schneider PF, editors. Proceedings of the International Workshop in Description Logics (DL'01); vol. 49 of *CEUR WS*. 2001,Stanford, CA, USA, August 1-3, 2001.

[63] Aiello M, Pratt-Hartmann I, van Benthem J, editors. Handbook of Spatial Logics. Springer; 2007.

[64] Keet CM. A formal theory of granularity. Phd thesis; KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy; 2008.

[65] Kutz O, Lutz C, Wolter F, Zakharyaschev M. E-connections of abstract description systems. Artificial Intelligence 2004;156(1):1–73.

[66] Cuenca Grau B, Parsia B, Sirin E. Combining OWL ontologies using ε-connections. Web Semantics: Science, Services and Agents on the World Wide Web 2006;4(1):40–59.

[67] Khan Z, Keet CM. The foundational ONtology SElection Tool ONSET. online; 2012. URL `http://www.meteck.org/files/onset/`.

[68] Baader F, Calvanese D, McGuinness DL, Nardi D, Patel-Schneider PF, editors. The Description Logics Handbook – Theory and Applications. Cambridge University Press; 2 ed.; 2008.

Algorithm 1 *PS Polygon to OWL*

Require: diagram is not empty

repeat

$x \leftarrow$ getPolygon() %% *select a polygon, and obtain information:*

shape $\leftarrow$ getPolygonShape(x)

colour $\leftarrow$ getPolygonColour(x)

label $\leftarrow$ getPolygonLabel(x)

if *label* consists of capitals and integers **then**

`name` $\leftarrow$ resolve abbreviation of *label*

%% "`name`" *is a variable that holds the name of the class that will be added to the ontology*

else

`name` $\leftarrow$ *label*

end if

associate x with `name`

%% *this is useful for the arrows in Algorithm 2*

select case

shape = oval **and** *colour* = red:

add `name` $\sqsubseteq$ `Protein` to the ontology

shape = oval **and** *colour* = purple:

add `name` $\sqsubseteq$ `NuclearReceptor` to the ontology

add `name` $\sqsubseteq \exists$`inheres_in.Protein` to the ontology

shape = rectangular **and** *colour* = yellow:

add `name` $\sqsubseteq$ `CellProcess` to the ontology

%% *Let us assume BioTop is imported as well*

parts $\leftarrow$ getPartonomy(Cell)

$y \leftarrow$ selectPart(*parts*)

add `name` $\sqsubseteq \exists$`located_in.y` to the ontology

shape = hexagon:

%% *and so forth for the other polygons*

end select case

search the PRO Protein Ontology

if `name` == `termX` in PRO **then**

add `name` $\equiv$ `termX` to the ontology

%% *with* `termX` *a variable denoting a term in PRO*

end if

until all shapes have been processed

%% *one can proceed with Algorithm 2*

Algorithm 2 *PS arrow to OWL*

Require: diagram has been processed by Algorithm 1

repeat

$x \leftarrow$ getArrow() %% *select an arrow, and obtain information:*

colour $\leftarrow$ getArrowColour(x)

$y \leftarrow$ getArrowBase(x)

$z \leftarrow$ getArrowHead(x)

%% *now obtain their respective class, thanks to Algorithm 1*

`namey` $\leftarrow retrieveClass(y)$

`namez` $\leftarrow retrieveClass(z)$

select case

colour = blue:

add `namez` $\sqsubseteq \exists$`expressed_by.namey` to the

to the ontology

colour = grey:

shape $\leftarrow$ getArrowShapeMiddle(x)

if *shape* = square **then**

add `namez` $\sqsubseteq \exists$`regulated_by.namey` to the

ontology

else %% *i.e., it is a cricle*

add `namez` $\sqsubseteq \exists$`regulated_directly_by.namey`

to the ontology

end if

colour = purple:

%% *and so forth for the other arrow colours*

end select case

search equivalence axioms with PRO

if `namey` and `namez` occur in an equivalence axioms and in `namez` $\sqsubseteq \exists$`OP.namey` with PRO **then**

remove axiom from ontology

%% *with* `OP` *a variable denoting an object property obtained also in PRO*

%% *this removes the redundant axioms*

end if

until all arrows have been processed