A Highly Literate Approach to Ontology Building

Phillip Lord and Jennifer D. Warrender

School of Computing Science, Newcastle University, Newcastle-upon-Tyne, UK

Abstract. Ontologies present an attractive technology for describing bio-medicine, because they can be shared, and have rich computational properties. However, they lack the rich expressivity of English and fit poorly with the current scientific "publish or perish" model. While, there have been attempts to combine free text and ontologies, most of these perform *post-hoc* annotation of text. In this paper, we introduce our new environment which borrows from literate programming, to allow an author to co-develop both text and ontological description. We are currently using this environment to document the Karyotype Ontology which allows rich descriptions of the chromosomal complement in humans. We explore some of the advantages and difficulties of this form of ontology development.

1 Introduction

Ontologies have been used extensively to describe many parts of bio-medicine. Ontologies have two key features which make their usage attractive. First, they provide a mechanism for standardizing and sharing the terms used in descriptions, making comparison easier and, secondly, they provide a computationally amenable semantics to these descriptions, making it possible to draw conclusions about the relationships between descriptions even when they share no terms in common.

Despite these advantages, the oldest and most common form of description in biology is free text. Free text has numerous advantages compared to ontologies: it is richly expressive, is widely supported by tooling, and while the form of language used in science ("Bad English" [16]) may not be easy to use, understand or learn, it is widely taught and most scientists are familiar with it.

Between these two extremes of computable amenability, there are a full array of different techniques. A "database" such as UniProt, for instance, appears to be highly structured but also contains a large quantity of "annotation" that appears to be free text; although, even this contains informal structure, which can be found and analysed by text analysis [1]. We can set this against descriptions of biological methods which appear in the form of a scientific paper. The two forms of description have largely been used independently. Ontology terms are used in semi-structured formats such as a UniProt record, or minimum information documents, but in general, ontology terms and the free text are in different parts of the record.

In this paper, we show how we can integrate ontological and textual knowledge in a single authoring environment, and describe how we are applying this to describing karyotypes.

2 Developing Knowledge

First, we ask the question, why is it difficult to relate ontological and textual descriptions during authoring. One possible explanation is that the two forms have very different "development environments"[1]. The main documentation environment used within science is Word, followed by LaTeX, common in more mathematical environments. More recently, there has also been interest in various light-weight markup languages, such as markdown. In the case of Word, the development environment is a single tool which (effectively) defines the file format, and the user interface that the author uses to interact with it; with both LaTeX and other markup languages, there is a tool chain in use, often with several options at each step, meaning that different authors have (somewhat) different environments.

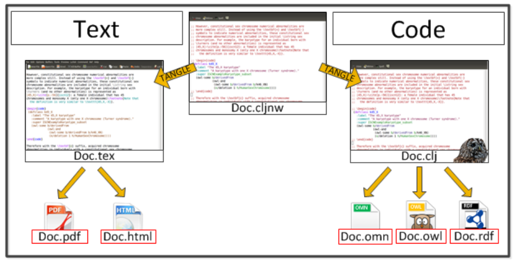
Ontology development environments also come in many different forms. Early versions of the Gene Ontology, for instance, used a bespoke text file format and a text editor – an approach rather similar to the light-weight markup languages of today. This had the significant advantage of a low-technological barrier to entry, at least for authors, as well as easy integration with tools such as version control systems which enabled collaborative working. It works poorly using XML native formats like OWL (Ontology Web Language), however. More modern environments, such as Protégé and OBO-Edit provide a much more graphical interface. These generally provide a much richer way of interacting with an ontology; authors can see whole terms at once, using a variety of syntaxes and allow rapid navigation through the class hierarchy, something which most ontology authors do a lot [13].

While these environments add a lot of value, they do not necessarily integrate well with text. Both Protégé and OBO-Edit have a class-centric view and are biased toward showing the various logical entities in the ontology, as opposed to the textual aspects. Indeed, this bias is shown even at the level of OWL. For example, annotations on an entity (or rather an axiom) are a *set* rather than a list, while ordering is generally considered to be essential for most documents.

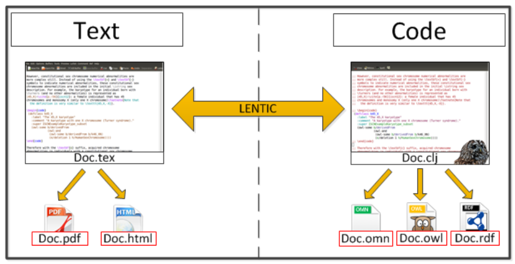
While there have been many attempts to integrate textual and ontological knowledge, these have mostly involved *post-hoc* annotation of ontological entities using text analysis. A notable exception to this is the Ontology Word add-in which uses text-analysis to suggest ontology terms that can be used to annotate text at the point of authorship [2].

With this divergence of development environments, it seems hard to understand how we could square the circle of combining text and ontology development. Next, we describe the Karyotype Ontology and how the novel development methodology we used for this ontology allows us to achieve this.

[1] We lack a good term which covers word-processor, editor and IDE.



(a) The traditional workflow



(b) The lenticular workflow

Fig. 1: Two workflows for the creation of literate ontologies. In a) the ontology is developed in an intermediate format (such as NoWeb), and the documentation and code-centric versions are generated. In b) both versions are developed simultaneously as views.

3 The Karyotype Ontology

A karyotype describes the number of chromosomes and any alterations from the normal. These are visible under the light microscope, and when stained have a characteristic banding pattern which can be used to distinguish between different chromosomes and the positions on these chromosomes. In humans, these alterations are described by their type, such as inversions, deletions or duplications and by their location, specified by a chromosome number and band number, following the *ISCN* specification. So, `46,XY,t(1;3)(p22;q13.1)` describes a male with a translocation from chromosome `1p22` to chromosome `3q13.1`. The Karyotype Ontology is, effectively, an ontological implementation of this ISCN specification for human karyotype nomenclature [10][2].

The Karyotype Ontology [15] was a challenging ontology to build because it is large but highly repetitive. It provided the original motivation for and has been developed with Tawny-OWL [8], our novel ontology environment which provides a fully programmatic development. Tawny-OWL is implemented as a Domain-Specific Language (or DSL) using the commodity Clojure language and inherits its programmatic capabilities directly from there. Simple ontological

[2] ISCN 2013 is now available

```
In \ko, each karyotype is modelled by explicitly
stating the base karyotype and any abnormality
events, using the |b/derivedFrom| and
|e/hasDirectEvent| relations respectively. For this
exemplar, the base karyotype is |k/46,XX|, as the
tumour originated from a female. In addition, we
model the |1| deletion abnormality using a
cardinality restriction and the |e/Deletion| and
|h/HumanChromosome22| classes.

\begin{code}
(defclass k45_XX_-22
 :label "The 45,XX,-22 karyotype"
 :comment "A karyotype with monosomy 22."
 :super ISCNExampleKaryotype_subset
 (owl-some b/derivedFrom b/k46_XX)
 (exactly 1 e/hasDirectEvent
          (owl-and e/Deletion
                   h/HumanChromosome22)))
\end{code}
```

(a) A document-centric view

```
;; In \ko, each karyotype is modelled by explicitly
;; stating the base karyotype and any abnormality
;; events, using the |b/derivedFrom| and
;; |e/hasDirectEvent| relations respectively. For this
;; exemplar, the base karyotype is |k/46,XX|, as the
;; tumour originated from a female. In addition, we
;; model the |1| deletion abnormality using a
;; cardinality restriction and the |e/Deletion| and
;; |h/HumanChromosome22| classes.

;; \begin{code}
(defclass k45_XX_-22
 :label "The 45,XX,-22 karyotype"
 :comment "A karyotype with monosomy 22."
 :super ISCNExampleKaryotype_subset
 (owl-some b/derivedFrom b/k46_XX)
 (exactly 1 e/hasDirectEvent
          (owl-and e/Deletion
                   h/HumanChromosome22)))
;; \end{code}
```

(b) The ontology-centric view

Fig. 2: Two lenticular views over an ontology source

statements can be written with a syntax inspired by OWL Manchester notation [3]; repetitive statements can be built automatically by writing functions which encapsulate and abstract over the simpler statements, a process we call "patternisation" [14]. Tawny-OWL can be used to generate any ontology and is not specific to the Karyotype Ontology; the latter however is our most extreme example of a patternised ontology with over 1000 classes using a single pattern. Tawny-OWL thus fulfils the requirement for efficient population of an ontology, something which tools like Protégé are lacking [12].

In addition to its programmatic capabilities, Tawny-OWL also allows the user to take advantage of commodity Clojure development environments. For instance, auto-complete, syntax highlighting, indentation and the REPL (Read-Eval-Print-Loop, essentially a shell) all comes direct from Clojure. In this way, we have managed to combine the advantages of text-based environments for editing ontologies i.e. the use of a standard editing environment and integration with version control, while maintaining (and in some ways surpassing) the power of tools like Protégé.

We consider next the implications that this has for the ability to integrate ontological and textual descriptions.

4 The Genesis of Literate Ontology

As Tawny-OWL is based on a full programming language, it supports a feature which at first seems quite inconsequential: comments. As with almost every programming language, it is possible to add free, unstructured text to the same source code that defines the ontology. While opinions vary on the role of comments in programmatic code, perhaps the most extreme is that of literate programming [4] which suggests that code should be usable both as a program capable of execution and as a document capable of reading.

A key aspect of literate programming is that neither view should have primacy, which separates it from much weaker systems such as, for example, JavaDoc, where the documentation very much fits into the code. We call this form of development *code-centric*. A more traditional approach uses *tangling*[3] – here a single source document contains both ontological and document source is created. It is then tangled to produce two forms of generated code which in turn compile into the executable and documentation form (see Figure 1a). This form of editing is used by a number of different systems, two of the most heavily used of which are DocTeX which uses LaTeX to document LaTeX[4] or Sweave [5] which combines LaTeX and R [9], the statistical programming language.

Our early attempts at literate ontology development used this approach. We tried embedding OWL into LaTeX [7]. As an alternative, we also build a system which allowed easy insertion of cross-references between a LaTeX file and Manchester OWL notation [6]. However, we found both to be highly-unusable. In one sense, tangling achieves the task of putting the executable and documentable sections of a code-base on an equal footing. However, in practice, there is a problem; the programmer has to edit the untangled form. These days programmers are used to extremely rich development environments which must be fully aware of the computational amenable nature of the source code to function. In both cases, our early experiments allowed the use of a LaTeX development environment, but provided a very weak ontology development environment similar to the early use of text editors. We call this form of development *document-centric*. We found this form of document-centric development so unattractive that it has been abandoned.

5 Literate Programming with Lenticular Views

The development of Tawny-OWL would make a tangling approach more viable, but still we must choose: a document-centric approach would involve editing Clojure source code without any IDE support (e.g. code evaluation, completion, as well as indentation or syntax highlighting for the Clojure sections) while a code-centric approach would lack support for LaTeX editing (e.g. citation insertion, cross-referencing as well as indentation or syntax-highlighting for the LaTeX sections).

Our latest solution attempts to square this circle. We provide a multi-view approach to editing, which allows the author to see her source code in either a *document-centric* or a *code-centric* view. We call this approach *lenticular* text, named after lenticular printing which produces images which change depending on your angle of viewing. This is an entirely novel approach to literate programming, effectively performing the tangling operation for the author as they type.

[3] The term "tangling" is not ours and is to our mind backward. However, it reflects the idea that source code is for consumption by a programmer and that this form is, therefore, untangled. The tangling process manipulates this clear form so that the computer can read it

[4] Which is genuinely as confusing as it sounds

A representation of the two views are shown in Figure 2. The two views, it should be noted, contain the same **text** but are syntactically different, such that the document-centric view is entirely valid LaTeX code, while the ontology-centric view is valid Tawny-OWL code.

We have now implemented lenticular text for the editor, Emacs[5], in a package called "lentic"[6]. We choose Emacs because it already provides a strong environment for editing both LaTeX and Clojure[7] A key feature of this implementation is that both views exist simultaneously in Emacs, and provide all the features of the appropriate development environment; for example, "tab-completion" works in both the document-centric view (completing LaTeX macros) and in the ontology-centric view (completing ontology identifiers). We can launch a compilation of the document-centric view (producing a PDF), or evaluate our ontology, perhaps reasoning over it, in the code-centric view. Therefore, we have achieved a key aim of literate programming: neither view holds primacy and the author can edit either. The overall workflow is shown in Figure 1b.

6 A Literate Karyotype

The ISCN which describes karyotypes is an informal specification, combined with many descriptions of particular karyotypes. For example, here we quote two examples from page 56, ISCN 2009. These examples help to define the specification further.

- **45,X** A karyotype with one X chromosome (Turner syndrome).
- **47,XYY** A karyotype with one X chromosome and two Y chromosomes (Klinefelter syndrome).

In the Karyotype Ontology, we have encoded many of these examples, partly to test that our ontology is capable of representing the ISCN specification. Through the use of lenticular text, we are able to annotate these descriptions both with references to the original work in ISCN as well as implementation notes, describing our representation. We are steadily converting the whole of the Karyotype Ontology into literate form; as an example of how this process works, we have included the output of part of the Karyotype Ontology at the end of this paper (see Section A). In short, the karyotype ontology is becoming a fully literate ontology.

7 Discussion

In this paper, we have described our methodology for integration of text and ontological statements at authoring time, using lenticular text to enable literate

[5] `https://www.gnu.org/software/emacs/`
[6] `https://github.com/phillord/lentic`
[7] It also relatively easy to extend, and has support for Manchester OWL Notation added by one of us (PL).

ontology development. This is a significant advance over, for example, the Word Ontology plug-in, which enables the use of ontology *annotation* at authoring time. With lenticular text, we are not limited to annotation with existing terms; we can define terms of arbitrary complexity, allowing us to post-coordinate our definitions [11].

The combination of Tawny-OWL and lenticular text is an extremely rich environment. We are aware, however, that it is a specialist environment. To develop a literate ontology the author needs: to use Tawny-OWL, program in Clojure, a Clojure development environment, write documents in LaTeX, and use lentic package which is Emacs-based. In reality, though, the tools described here are not tightly coupled. In particular:

– Clojure programming is only needed to extend Tawny-OWL.
– Clojure is not tied to Emacs; there are other, well-supported environments.
– Currently, lenticular text is novel and only implemented by the Emacs lentic package but it could be implemented in other environments[8]
– It is possible to edit a literate ontology *without* using lenticular views, effectively replicating the traditional tangling workflow (see Figure 1a)[9].
– Neither lenticular text nor the lentic package is specific to LaTeX or Tawny-OWL [10].
– Both lenticular text and the lentic package are useful for general purpose programming and are not ontology specific[11].
– Other embedded DSLs for OWL exist, such as ScOWL[12] and OWLJS[13].

While, we accept that the adoption of all the tooling described in the paper maybe be relevant to very few developers, the use of parts of it have much more widespread utility. It is, of course, unlikely to overtake Word as the main tool for scientific authoring, it does have the potential to fulfil a distinct niche as Sweave has done for statisticians.

We have, however, hit some problems with this process. We would like to have developed the Karyotype Ontology alongside the text from ISCN, so that the justification for each of the statements we have made would be clear. Unfortunately, the ISCN is published under a non-permissive licence which prevents the production of this sort of derived work. It is not even possible to hyperlink through to the relevant sections of ISCN, as it is released only on paper. The

[8] The first simple, version of lentic was around 1k loc, so this is not challenging to implement. Later versions are larger, as making the implementation efficient and scalable is somewhat harder.
[9] We actually use Lentic and Emacs in "batch" for this purpose, but an independent tool could be implemented very easily
[10] Currently, lentic supports various combinations of Emacs- Lisp, Haskell or Clojure, with asciidoc, org-mode or LaTeX.
[11] Lentic is self-documenting using Emacs-Lisp and org-mode, and Tawny-OWL is being converted. We also have entirely non-ontological users
[12] `https://github.com/phenoscape/scowl`
[13] `https://github.com/cmungall/owljs`

irony of our attempt to use Semantic Web technology on a resource that has not even reached the web has not escaped our notice.

Likewise, our use of LaTeX integrates poorly with the web. While it is possible to turn LaTeX source into HTML, it is not straight-forward. Lentic supports other formats which are more suitable for this purpose (org-mode and asciidoc) although they are formats aimed a programmers and have, for example, comparatively weaker support for literate referencing. We also currently have little support for cross-referencing *between* the forms – so referring to ontology terms in text, for example, or sections in the documentation from within ontology `rdfs:comment` annotations. We believe that these extensions are entirely achievable in future.

Still, there are many other potential biomedical uses[14] for this form of technology, beyond karyotype descriptions. We are currently also investigating clinical guidelines which describe treatment plans – fortunately in the UK, these are published with a permissive license. In these cases, the knowledge being reproduced is such high value and expensive to produce that the costs imposed by adding semantics in a specialist environment are probably worthwhile. With Tawny-OWL and lentic, we now have tools available which allow us to achieve this goal.

Acknowledgements

This work was supported by Newcastle University.

References

1. Bell, M.J.: Provenance, Propagation and Quality of Biological Annotation. Ph.D. thesis, Newcastle University (2014)
2. Fink, J.L., Fernicola, P., Chandran, R., Parastatidis, S., Wade, A., Naim, O., Quinn, G.B., Bourne, P.E.: Word add-in for ontology recognition: semantic enrichment of scientific literature. BMC Bioinformatics 11(1), 103 (2010), `http://dx.doi.org/10.1186/1471-2105-11-103`
3. Horridge, M., Patel-Schneider, P.F.: Owl 2 web ontology language manchester syntax (second edition). Tech. rep. (2012), `http://www.w3.org/TR/owl2-manchester-syntax/`
4. Knuth, D.E.: Literate programming. The Computer Journal 27, 97–111 (1984)
5. Leisch, F.: Sweave: Dynamic generation of statistical reports using literate data analysis. In: Härdle, W., Rönz, B. (eds.) Compstat 2002 — Proceedings in Computational Statistics. pp. 575–580. Physica Verlag, Heidelberg (2002), `http://www.stat.uni-muenchen.de/~leisch/Sweave`, iSBN 3-7908-1517-9
6. Lord, P.: Literate omn. `http://www.russet.org.uk/blog/1213` (2009), `\url{http://www.russet.org.uk/blog/1213}`
7. Lord, P.: Literate owl (well on blogs). `http://www.russet.org.uk/blog/1204` (2009), `http://www.russet.org.uk/blog/1204`

[14] As well as outside biomedicine: perhaps inenvitably, we have also used it to describe pizza.

8. Lord, P.: The Semantic Web takes Wing: Programming Ontologies with Tawny-OWL. http://arxiv.org/abs/1303.0213 (2013), http://arxiv.org/abs/1303.0213
9. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2014), http://www.R-project.org
10. Shaffer, L., Slovak, M., Campbell, L. (eds.): ISCN 2009: An International System for Human Cytogenetic Nomenclature (2009). Karger (2009), http://books.google.co.uk/books?id=z0yNPgAACAAJ
11. Stevens, R., Sattler, U.: Post-coordination: Making things up as you go along. http://ontogenesis.knowledgeblog.org/1305 (2013), http://ontogenesis.knowledgeblog.org/1305
12. Vigo, M., Bail, S., Jay, C., Stevens, R.: Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design. International Journal of Human-Computer Studies 72(12), 835–845 (Dec 2014), http://dx.doi.org/10.1016/j.ijhcs.2014.07.005
13. Vigo, M., Jay, C., Stevens, R.: Protege4us: harvesting ontology authoring data with protege. In: The Semantic Web: ESWC 2014 Satellite Events, pp. 86–99. Springer (2014)
14. Warrender, J.D., Lord, P.: A pattern-driven approach to biomedical ontology engineering. SWAT4LS 2013 (2013)
15. Warrender, J.D., Lord, P.: The Karyotype Ontology: a computational representation for human cytogenetic patterns. Bio-Ontologies 2013 (2013)
16. Wood, A., Flowerdew, J., Peacock, M.: International scientific english: The language of research scientists around the world. Research Perspectives on English for Academic Purposes pp. 71–83 (2001), http://dx.doi.org/10.1017/CBO9781139524766.008

A Appendix: What is an ISCN String?

This section[1] provides a lenticular review of how ISCN Strings are defined by the specification and are modelled using The Karyotype Ontology, by focusing on a subset of exemplars defined in the ISCN.

```
;; Define namespace
(ns ^{:doc "Defining example karyotypes from the ISCN2013."
      :author "Jennifer Warrender"}
  ncl.karyotype.iscnexamples_subset
  (:use [tawny.owl])
  (:require [ncl.karyotype
             [karyotype :as k]
             [human :as h]
             [events :as e]
             [base :as b]]))

;; Define ontology
(defontology iscnexamples_subset
  :iri
  "http://www.purl.org/captau/karyotype/iscnexamples_subset"
  :prefix "iexs:"
  :comment "Subset of the ISCN Example Karyotypes ontology
  for Human Karyotype Ontology, written using the Tanwy_OWL
  library.")

;; Import all karyotype axioms
(owl-import k/karyotype)

;; Create a new subclass of Karyotype
(defclass ISCNExampleKaryotype_subset
  :super k/Karyotype)
```

In The Karyotype Ontology "normal" karyotypes for each ploidy level are modelled in the `base` ontology; thus we import all associated axioms into the current ontology.

```
(owl-import b/base)
```

However, not all karyotypes are normal; they can include a variety of abnormalities. There are two types of abnormality. *Numerical abnormalities* are abnormalities that affect the number of chromosomes present in the karyotype, either by gaining or losing whole chromosomes. *Structural abnormalities* are abnormalities that involve only parts of the chromosomes[2].

In order to model karyotypes, we need concepts in the ontology that model the human chromosomes and the numerical abnormality events. These are mod-

[1] This section is a demonstration of the output from our lenticular representation of karyotypes. It should not be considered to be a formal part of the paper.

[2] For simplicity, structural abnormalities will not be discussed at this time.

elled in the `human` and `events` ontologies respectively; thus we import all axioms from both.

```
(owl-import e/events)
(owl-import h/human)
```

In the ISCN, numerical abnormalities are represented in the ISCN String using symbols and abbreviated terms. For numerical abnormalities, the symbol `-` is used to represent the loss of chromosomes while + represents the gain of chromosomes.

For example, the karyotype of a female individual that has lost one chromosome `22` (and no other abnormalities) is represented as `k45,XX,-22` [1, p. 57]; this results in 45 chromosomes and monosomy (one copy of) chromosome `22`.

In The Karyotype Ontology, each karyotype is modelled by explicitly stating the base karyotype and any abnormality events, using the `b/derivedFrom` and `e/hasDirectEvent` relations respectively. For this exemplar, the base karyotype is `k/46,XX`, as the tumour originated from a female. In addition, we model the `1` deletion abnormality using a cardinality restriction and the `e/Deletion` and `h/HumanChromosome22` classes. However due to the programmatic nature of Tawny-OWL, we can implement parameterised patterns [14], thus simplifying the deletion abnormality definition to one line of code, using the `e/deletion` pattern.

```
(defclass k45_XX_-22
  :label "The 45,XX,-22 karyotype"
  :comment "A karyotype with monosomy 22."
  :super ISCNExampleKaryotype_subset
  (owl-some b/derivedFrom b/k46_XX)
  (e/deletion 1 h/HumanChromosome22))
```

Similarly, the karyotype of a tumour from a female individual that has lost one chromosome `X` (and no other abnormalities) is represented as `k45,X,-X` [1, p. 56]. In The Karyotype Ontology, this karyotype is modelled with the base karyotype `b/46,XX` and `1` deletion event that involves `h/HumanChromosomeX`.

```
(defclass k45_X_-X
  :label "The 45,X,-X karyotype"
  :comment "A tumor karyotype in a female with loss of one X
  chromosome."
  :super ISCNExampleKaryotype_subset
  (owl-some b/derivedFrom b/k46_XX)
  (e/deletion 1 h/HumanChromosomeX))
```

However, the classification of abnormalities is not so simple; an abnormality can be also classified as either a *constitutional* or *acquired* abnormality[3]. A constitutional abnormality, also known as an in-born abnormality, is an abnormality that is present in (almost) all cells of an individual and exists at the earliest stages of embryogenesis, while an acquired abnormality is an abnormality that develops in somatic cells [2].

[3] All previous exemplars define acquired abnormalities.

Generally, constitutional abnormalities are indicated using the suffix **c**. For example the ISCN String `46,XY,+21c,-21` [1, p. 58] represents the karyotype of tumour cells taken from a male individual, that had a constitutional trisomy `21` and has acquired disomy `21`. Using this representation we see that karyotypes with constitutional abnormalities explicitly define two types of canonicalisation; one of the individual and the other for the cell line they have given rise to.

In The Karyotype Ontology, constitutional abnormalities are also modelled explicitly using the `e/hasDirectEvent` relation. However unlike acquired abnormalities, constitutional abnormalities are modelled as a nested restriction in conjunction with the base karyotype. In this exemplar:

- the base karyotype is `b/46,XY` (as the karyotype originates from a male individual).
- the `1` constitutional abnormality is a gain of one chromosome `21`. The associated parameterised pattern for gain is `e/addition`.
- the `1` acquired abnormality is a loss of one chromosome `21`.

```
(defclass k46_XY_+21c_-21
  :label "The 46,XY,+21c,-21 karyotype"
  :comment "Acquired loss of one chromosome 21 in a patient
  with Down syndrome."
  :super ISCNExampleKaryotype_subset
  ;;aka 47,XY,+21
  (owl-some b/derivedFrom
           (owl-and
            (owl-some b/derivedFrom b/k46_XY)
            (e/addition 1 h/HumanChromosome21)))
  (e/deletion 1 h/HumanChromosome21))
```

However, constitutional sex chromosome numerical abnormalities are more complex still. Instead of using the + and - symbols to indicate numerical abnormalities, these constitutional sex chromosome abnormalities are included in the initial ISCN String sex description. For example, the karyotype for an individual born with Tuners Syndrome (and no other abnormalities) is represented as `45,X` [1, p. 56]: a female individual that has 45 chromosomes and monosomy X (only one X chromosome)[4].

```
(defclass k45_X
  :label "The 45,X karyotype"
  :comment "A karyotype with one X chromosome (Turner
  syndrome)."
  :super ISCNExampleKaryotype_subset
  (owl-some b/derivedFrom
           (owl-and
            (owl-some b/derivedFrom b/k46_XN)
            (e/deletion 1 h/HumanSexChromosome))))
```

[4] Note that the definition is very similar to `45,X,-X`.

With the **c** suffix, acquired chromosome abnormalities in individuals with a constitutional sex chromosome abnormality can easily be distinguished. For example the ISCN String `46,Xc,+21` [1, p. 57] represents tumour cells taken from a female individual with Tuners Syndrome; a constitutional monosomy X and an acquired trisomy 21.

```
(defclass k46_Xc_+21
  :label "The 46,Xc,+21 karyotype"
  :comment "Tumor cells with an acquired extra chromosome 21
  in a patient with Turner syndrome."
  :super ISCNExampleKaryotype_subset
  ;;aka 45,X
  (owl-some b/derivedFrom
            (owl-and
             (owl-some b/derivedFrom b/k46_XN)
             (e/deletion 1 h/HumanSexChromosome)))
  (e/addition 1 h/HumanChromosome21))

;; Implement disjoint axioms
(as-disjoint k45_XX_-22 k45_X_-X
             k46_XY_+21c_-21 k45_X k46_Xc_+21)
```

Now that we defined a few exemplar karyotypes, we discuss the definition of sex.

A.1 Defining Sex

While building this ontology, we found that sex is not as intuitive as it seems. The obvious definition for sex was that a "male" karyotype should be defined as a karyotype with a `Y` chromosome, while a "female" karyotype as one without. However further investigation showed that these definitions are, in fact, too simplistic as the karyotype `45,X,-Y`[5], has no `Y` chromosome, yet would generally be considered to be a "male" karyotype.

Therefore, the finalised definition for sex, as shown below considers the history of the karyotype by asserting a `derivedFrom` relation[6]. Using these definitions, the `45,X,-Y` karyotype can be correctly stated as being a "male" karyotype.

```
(defclass MaleKaryotype
  :equivalent
  (owl-or
   b/k46_XY
   (owl-some b/derivedFrom b/k46_XY)))

(defclass FemaleKaryotype
  :equivalent
  (owl-or
```

[5] A male-derived cell line which has lost its `Y` chromosome.

[6] Due to the transitive property of `b/derivedFrom`, we can also determine the sex of karyotypes that contain constitutional abnormalities.

```
b/k46_XX
(owl-some b/derivedFrom b/k46_XX)))
```

However these definitions are unable to ontologically categorise the `45,X` karyotype as either female or male though it would generally be considered a "female" karyotype. There is no correct answer to this problem. We could either redefine our female karyotype to include the `45,X` karyotype or add phenotypic sex. This decision needs to be taken by the domain experts themselves.

References

1. L. Shaffer, M.-J. J., and S. M., editors. *ISCN 2013: An International System for Human Cytogenetic Nomenclature (2013)*. Karger, 2012.
2. M. M. Wintrobe and J. P. Greer. Wintrobe's clinical hematology. 1, 2009.