Sequences in OWL

Nick Drummond, A. Rector, G. Moulton, Robert Stevens, M. Horridge, H. Wang, J Seidenberg
Sequences in OWL
Nick Drummond, A. Rector, G. Moulton, Robert Stevens, M. Horridge, H. Wang, J Seidenberg

Or “Making reasoners work hard”
Overview
Overview

- Presentation AND (hasContents SOME UseCases) AND
- (hasNext SOME (Presentation AND (hasContents SOME ListTheory) AND
- (hasNext SOME (Presentation AND (hasContents SOME RDFLists) AND
- (hasNext SOME (Presentation AND (hasContents SOME OWLLLists) AND
- (hasNext SOME (Presentation AND (hasContents SOME ProteinSequences) AND
- (hasNext SOME (Presentation AND (hasContents SOME AdvantagesAndDisadvantages) AND
- (hasNext SOME (Presentation AND (hasContents SOME Conclusion) AND
- (hasNext SOME EmptyList))))))))))))
Lists: Use Cases
Lists: Use Cases

Time related events

- Spawner
- Egg
- Adult
- Alevin
- Smolt
- Fry
Lists: Use Cases

Time related events

Conceptually linked structures
Lists: Use Cases

Time related events

Physically linked structures

Conceptually linked structures
Lists Theory
Lists Theory

- OWL & RDF Lists use the same idea
Lists Theory

- OWL & RDF Lists use the same idea
- A head, followed by by a tail (sublist)
Lists Theory

- OWL & RDF Lists use the same idea
- A head, followed by by a tail (sublist)

A   B   C   nothing
List Theory

- OWL & RDF Lists use the same idea
- A head, followed by by a tail (sublist)

A B C nothing
Lists Theory

- OWL & RDF Lists use the same idea
- A head, followed by by a tail (sublist)
Lists Theory

- OWL & RDF Lists use the same idea
- A head, followed by by a tail (sublist)
Lists Theory

- OWL & RDF Lists use the same idea
- A head, followed by by a tail (sublist)
Lists Theory

- OWL & RDF Lists use the same idea
- A head, followed by a tail (sublist)
RDF Lists

A → B → C
Constructs are:

A → B → C
RDF Lists

Constructs are:

- **rdf:list** – the list itself (can be thought of as the list element)
RDF Lists

Constructs are:

- **rdf:list** – the list itself (can be thought of as the list element)
- **rdf:first** – a pointer to the head of the list
RDF Lists

Constructs are:

- **rdf:list** – the list itself (can be thought of as the list element)
- **rdf:first** – a pointer to the head of the list
- **rdf:rest** – a pointer to the tail (sublist) containing the other elements
RDF Lists

Constructs are:
- `rdf:list` – the list itself (can be thought of as the list element)
- `rdf:first` – a pointer to the head of the list
- `rdf:rest` – a pointer to the tail (sublist) containing the other elements
RDF Lists

Constructs are:
- **rdf:list** – the list itself (can be thought of as the list element)
- **rdf:first** – a pointer to the head of the list
- **rdf:rest** – a pointer to the tail (sublist) containing the other elements
- **rdf:nil** – an rdf:list containing no other elements (terminator)
RDF Lists: Example
RDF Lists: Example

- The closed list ABC:
RDF Lists: Example

‣ The closed list ABC:
RDF Lists: Example

- The closed list ABC:

  rdf:list ( rdf:first A)
RDF Lists: Example

‣ The closed list ABC:

```
rdf:list (  
  rdf:rest rdf:list (  
    rdf:first A
  )
)  
```

```
  rdf:rest rdf:list (  
    rdf:first B
  )
```
RDF Lists: Example

» The closed list ABC:

```xml
<rdf:list>
  <rdf:first>A</rdf:first>
  <rdf:rest>
    <rdf:list>
      <rdf:first>B</rdf:first>
      <rdf:rest>
        <rdf:list>
          <rdf:first>C</rdf:first>
        </rdf:list>
      </rdf:rest>
    </rdf:list>
  </rdf:rest>
</rdf:list>
```
RDF Lists: Example

The closed list ABC:

```
rdf:list (  rdf:rest rdf:list (  rdf:rest rdf:list (  rdf:rest rdf:nil)))
```

```xml
<rdf:list rdf:rest= rdf:list (  rdf:rest rdf:list (  rdf:rest rdf:nil)))
```
RDF Lists: Example

The closed list ABC:

```xml
<rdf:list>
  <rdf:first>A</rdf:first>
  <rdf:rest>
    <rdf:list>
      <rdf:first>B</rdf:first>
      <rdf:rest>
        <rdf:list>
          <rdf:first>C</rdf:first>
          <rdf:rest rdf:nil/>
        </rdf:list>
      </rdf:rest>
    </rdf:list>
  </rdf:rest>
</rdf:list>
```
RDF Lists: Example

» The closed list ABC:

\[
\text{rdf:list ( rdf:first A }
\text{ rdf:rest rdf:list ( rdf:first B }
\text{ rdf:rest rdf:list ( rdf:first C }
\text{ rdf:rest rdf:nil }))}
\]

» Without the nil, this would be an open list

ie one that starts with ABC
OWL Ordering Support
OWL Ordering Support

- There is no inbuilt ordering in OWL
OWL Ordering Support

- There is no inbuilt ordering in OWL
- Ordering of statements in an OWL document has no semantics in OWL
OWL Ordering Support

- There is no inbuilt ordering in OWL
- Ordering of statements in an OWL document has no semantics in OWL
- Why not just use RDF statements?
OWL Ordering Support

- There is no inbuilt ordering in OWL
- Ordering of statements in an OWL document has no semantics in OWL
- Why not just use RDF statements?
  - not allowed in OWL-DL (because RDF is used in OWL serialisation)
OWL Ordering Support

- There is no inbuilt ordering in OWL
- Ordering of statements in an OWL document has no semantics in OWL
- Why not just use RDF statements?
  - not allowed in OWL-DL (because RDF is used in OWL serialisation)
  - we cannot reason with the lists we have modelled
OWL Ordering Support

- There is no inbuilt ordering in OWL
- Ordering of statements in an OWL document has no semantics in OWL
- Why not just use RDF statements?
  - not allowed in OWL-DL (because RDF is used in OWL serialisation)
  - we cannot reason with the lists we have modelled
- But OWL can model everything right? :)

© 2006, The University of Manchester
OWL Lists: Constructs
OWL Lists: Constructs

- class OWLLList isFollowedBy only OWLLList
OWL Lists: Constructs

- class OWLLList isFollowedBy only OWLLList

class EmptyList
   hasContents max 0
OWL Lists: Constructs

- class OWLList isFollowedBy only OWLList
  class EmptyList
    hasContents max 0
    not hasNext some owl:Thing
OWL Lists: Constructs

- class OWLLList isFollowedBy only OWLLList

class EmptyList
    hasContents max 0
    not hasNext some owl:Thing
OWL Lists: Constructs

- class OWLLList isFollowedBy only OWLLList
- class EmptyList
  hasContents max 0
  not hasNext some owl:Thing

- object property hasContents functional
OWL Lists: Constructs

- class OWLLList isFollowedBy only OWLLList
  class EmptyList
    hasContents max 0
    not hasNext some owl:Thing

- object property hasContents functional
OWL Lists: Constructs

- class OWLLList isFollowedBy only OWLLList

  class EmptyList
  hasContents max 0
  not hasNext some owl:Thing

- object property hasContents functional

- object property isFollowedBy transitive, range OWLLList

  object property hasNext functional
OWL Lists: Constructs

- class OWLLList isFollowedBy only OWLLList

- class EmptyList
  - hasContents max 0
  - not hasNext some owl:Thing

- object property hasContents functional

- object property isFollowedBy transitive, range OWLLList
- object property hasNext functional
OWL Lists: Constructs

- class OWLList isFollowedBy only OWLList

- class EmptyList hasContents max 0
  not hasNext some owl:Thing

- object property hasContents functional

- object property isFollowedBy transitive, range OWLList
  object property hasNext functional

- NB domain of properties is OWLList
OWL Lists: Constructs

A — B — C
Constructs are:

A → B → C
Constructs are:

- **OWLLList** – the list itself (can be thought of as the list element)
Constructs are:

- **OWLList** – the list itself (can be thought of as the list element)
- **hasContents** – a pointer to the head of the list
Constructs are:
- **OWLList** – the list itself (can be thought of as the list element)
- **hasContents** – a pointer to the head of the list
- **hasNext** – a pointer to the tail (sublist) containing the other elements
OWL Lists: Constructs

Constructs are:

- **OWLList** – the list itself (can be thought of as the list element)
- **hasContents** – a pointer to the head of the list
- **hasNext** – a pointer to the tail (sublist) containing the other elements
Constructs are:

- **OWLLList** – the list itself (can be thought of as the list element)
- **hasContents** – a pointer to the head of the list
- **hasNext** – a pointer to the tail (sublist) containing the other elements
- **EmptyList** – an OWLLList containing no other elements (terminator)
OWL Lists: Constructs

Constructs are:
– **OWLList** – the list itself (can be thought of as the list element)
– **hasContents** – a pointer to the head of the list
– **hasNext** – a pointer to the tail (sublist) containing the other elements
– **EmptyList** – an OWLList containing no other elements (terminator)
– **isFollowedBy** - transitive hasNext - for inferring indirect following elements
OWL Lists: Example
OWL Lists: Example

› The closed list ABC:
OWL Lists: Example

- The closed list ABC:
OWL Lists: Example

- The closed list ABC:
OWL Lists: Example

- The closed list ABC:

  List and
  hasContents some A and
OWL Lists: Example

- The closed list ABC:

List and
hasContents some A and
hasNext some ( List and
• The closed list ABC:

List and
hasContents some A and
hasNext some ( List and
hasContent some B and
OWL Lists: Example

The closed list ABC:

List and
hasContents some A and
hasNext some ( List and

hasContent some B and
hasNext some ( List and
The closed list ABC:

List and
hasContents some A and
hasNext some (
    List and
    hasContent some B and
    hasNext some (    List and
    hasContents some C and
OWL Lists: Example

The closed list ABC:

List and
hasContents some A and
hasNext some (List and
hasContent some B and
hasNext some (List and
hasContents some C and
hasNext some EmptyList)))
OWL Lists: Example

The closed list ABC:

```
List and
hasContents some A and
hasNext some (
    List and
    hasContent some B and
    hasNext some (List and
        hasContents some C and
        hasNext some EmptyList))
```
OWL Lists: Example

- The closed list ABC:

  List and
  hasContents some A and
  hasNext some ( List and
                  hasContent some B and
                  hasNext some ( List and
                              hasContents some C and
                              hasNext some EmptyList))

- remember isFollowedBy relations can be inferred
OWL Lists: Example

The closed list ABC:

List and
hasContents some A and
hasNext some (List and
  hasContent some B and
  hasNext some (List and
    hasContents some C and
    hasNext some EmptyList)))

remember isFollowedBy relations can be inferred
Eg ABC - isFollowedBy some (List and hasContents some C)
OWL Lists: Expressivity
OWL Lists: Expressivity

- What can we express?
OWL Lists: Expressivity

- What can we express?
  - seqA exactly matches seqB
OWL Lists: Expressivity

‣ What can we express?
  ‣ seqA exactly matches seqB
  ‣ seqA contains seqB
OWL Lists: Expressivity

- What can we express?
  - seqA exactly matches seqB
  - seqA contains seqB
  - seqA starts with seqB, seqA ends with seqB
OWL Lists: Expressivity

What can we express?

- seqA exactly matches seqB
- seqA contains seqB
- seqA starts with seqB, seqA ends with seqB
- seqA only contains elements of type A
OWL Lists: Expressivity

What can we express?

- seqA exactly matches seqB
- seqA contains seqB
- seqA starts with seqB, seqA ends with seqB
- seqA only contains elements of type A
- seqA is all As followed by seqB
OWL Lists: Expressivity

- What can we express?
  - seqA exactly matches seqB
  - seqA contains seqB
  - seqA starts with seqB, seqA ends with seqB
  - seqA only contains elements of type A
  - seqA is all As followed by seqB
  - seqA does not start/end/contain seqB
OWL Lists: Expressivity

What can we express?

- seqA exactly matches seqB
- seqA contains seqB
- seqA starts with seqB, seqA ends with seqB
- seqA only contains elements of type A
- seqA is all As followed by seqB
- seqA does not start/end/contain seqB
- seqA is seqB followed by anything, followed by seqC
OWL Lists: Expressivity

- What can we express?
  - seqA exactly matches seqB
  - seqA contains seqB
  - seqA starts with seqB, seqA ends with seqB
  - seqA only contains elements of type A
  - seqA is all As followed by seqB
  - seqA does not start/end/contain seqB
  - seqA is seqB followed by anything, followed by seqC
  - combinations of above
OWL Lists: Expressivity

- What can we express?
  - seqA exactly matches seqB
  - seqA contains seqB
  - seqA starts with seqB, seqA ends with seqB
  - seqA only contains elements of type A
  - seqA is all As followed by seqB
  - seqA does not start/end/contain seqB
  - seqA is seqB followed by anything, followed by seqC
  - combinations of above
    - eg seqA starts with seqB and ends with seqC
OWL Lists: Expressivity

- What can we express?
  - seqA exactly matches seqB
  - seqA contains seqB
  - seqA starts with seqB, seqA ends with seqB
  - seqA only contains elements of type A
  - seqA is all As followed by seqB
  - seqA does not start/end/contain seqB
  - seqA is seqB followed by anything, followed by seqC
  - combinations of above
    - eg seqA starts with seqB and ends with seqC
  - Elements can be any arbitrary OWL expression
OWL Lists: Expressivity

- What can we express?
  - seqA exactly matches seqB
  - seqA contains seqB
  - seqA starts with seqB, seqA ends with seqB
  - seqA only contains elements of type A
  - seqA is all As followed by seqB
  - seqA does not start/end/contain seqB
  - seqA is seqB followed by anything, followed by seqC
  - combinations of above
    - eg seqA starts with seqB and ends with seqC
  - Elements can be any arbitrary OWL expression
    - unions, restrictions etc
Motivating case: Protein Seqs
Motivating case: Protein Seqs

F26_YEAST Fructose-2,6-bisphosphatase:
Motivating case: Protein Seqs

F26_YEAST Fructose-2,6-bisphosphatase:
MGYSTISNDNDIKVCVICMVGLPARGKSFIISQKIIYRYSWLF
SIKAKCFNVGNYRRDVSGNVGPMDAEFFNFDNFDNFKLRELAAQNAIKDIVNFHTKEDGSVAVFVDFATNSTRKRRKKWLKDICEKNNIQPMFLESWSNDHELIIINNAKDIGSTSPDYENSEPHVAEADFLERIRQYERFYEPLDQKDKDMDTFIKLVNIIEEVVINKIRITYLESRIVFYVMNIRPKPKYIWLRSRHGE
SIYNVEKKGDSLSERGFQYAKKLEQLVKESEAGEINLTVWTSTLKLRTQQTANYLPYKQLKWKLALDELDAVGCDGMTYEEIEKEYPDKARNDKDKEYERYRGRGSEYRVDVIRLEQVIMELERQENVLIITHQAVLRICYAYFMNVPQEEPSWMSIPLHTLILKLEPRAYGTKVTKIKANIPAVSTYKEKGTSCQGELESQSSTKLHQLLNDSPLEDKFA
Motivating case: Protein Seqs
Motivating case: Protein Seqs

- Biologists want to classify proteins by recognising parts of the sequence that are similar to parts of other sequences
Motivating case: Protein Seqs

- Biologists want to classify proteins by recognising parts of the sequence that are similar to parts of other sequences
- Identifiable parts of the sequence are motifs
Motifs (Pattern Matching)
Motifs (Pattern Matching)

- An identifiable part of a protein sequence
Motifs (Pattern Matching)

- An identifiable part of a protein sequence
- A simple example:
  \[[IV]-A-[VI]-F-D-A-T-N-[TS]-T-[RK]-[EDK]-R-R-[HSDARK]\]
  that is:
  \[[Isoleucine or Valine], then Alanine, then [Isoleucine or Valine], then Phenylalanine \ldots\text{etc}\]
Motifs (Pattern Matching)

- An identifiable part of a protein sequence
- A simple example:
  \[[IV]-A-[VI]-F-D-A-T-N-[TS]-T-[RK]-[EDK]-R-R-[HSDARK]\]
  that is:
  \[[\text{Isoleucine or Valine}], \text{then Alanine, then} [\text{Isoleucine or Valine}], \text{then Phenylalanine} \ldots \text{etc}\]
- It is simple to model the \([XY]\) elements as OWL unions
Motifs (Pattern Matching)

- An identifiable part of a protein sequence
- A simple example:
  
  \[\text{[IV]-A-[VI]-F-D-A-T-N-[TS]-T-[RK]-[EDK]-R-R-[HSDARK]}\]

  that is:

  \([\text{Isoleucine or Valine}], \text{then Alanine, then [Isoleucine or Valine], then Phenylalanine \ldots etc} \]

- It is simple to model the \([XY]\) elements as OWL unions
- But there are other things also worth investigating...
Underspecification of Patterns
Underspecification of Patterns

- Motifs reflect the current level of knowledge and may, in cases, be over-constrained
Underspecification of Patterns

- Motifs reflect the current level of knowledge and may, in cases, be over-constrained
- In OWL we have more control over how we express the pattern
Underspecification of Patterns

- Motifs reflect the current level of knowledge and may, in cases, be over-constrained.
- In OWL we have more control over how we express the pattern.

(Polar and Tiny)-Amino-(Large and Positive)

[Ser Thr]-x-[Lys Arg]

(Ser, Gly, Lys)  (Thr, Gly, Arg)

biologist's abstraction (x is "any")

underspecified motif

concrete motifs
Underspecification of Patterns

- Motifs reflect the current level of knowledge and may, in cases, be over-constrained
- In OWL we have more control over how we express the pattern

By relaxing the description, we might find additional potential matches that could be investigated
Fingerprints (sets of motifs)
Fingerprints (sets of motifs)

Match a set of motifs (in order) in a sequence:

6PFRUCTKNASE fingerprint:

Motif 1 = [IV]-A-[VI]-F-D-A-T-N-[TS]-T-[RK]-[EDK]-R-R-[HSDARK]
Motif 2 = [KRQ]-[TVCPA]-[FLM]-F-[IVL]-E-S-[IVW]-[CS]-[DVN]-D-[PH]-[GDEAT]-[IVL]-[IV]
Motif 3 = P-D-Y-[KEVIPT]-[GNDE]-[CRLSK]-[NDHME]-[PTSQR]-[AEDGH]-[VSEKN]-[AVQS]-[AELTM]-[AKED]-[DE]-[FD]
Motif 4 = [VI]-[QR]-[DGT]-[HYF]-[IVL]-[QEA]-S-[RQK]-[ITAV]-[VA]-[YF]-[YF]-[LV]-[ML]-N-[ITF]-[HRN]-[VPL]-[QTHLKA]-[PD]-[RK]-[TSAYQ]
Motif 5 = I-[YW]-[LI]-[CST]-R-[HS]-G-[EQ]-[NS]-[IEQ]-[HYLFD]-N-[VLAi]-[QRLKMES]-[GK]-[RK]-[IL]-G-G-[DN]-[SPA]-[GPSH]-L
Motif 6 = A-G-[VID]-[CY]-[ED]-[EG]-[LM]-T-Y-[EA]-[ED]-I-[RQEK]-[DKEQN]-[THRQEN]-[YF]-P
Matching

F26_YEAST Fructose-2,6-bisphosphatase:
MGYSTISNDNIKVCVIMVGLPARGKSFISQKIIRYLSSLWLSIKAKCFNVGVNYYRRDVGSGNVPMDAEFFNFENDNFKLERLAAQQNAIKDIVNFTKEDGSVAVFDATNSTRKRRKWLIKDICEKNNIQPFLESWSNDHELIIINNAKDIGSTSPDYENSEPHVAEADFRLERIRQYERFYEPLDPQKDKDAGTFIKLVNIIIEVVINKIRTYLESRIVFYVMNIRPKPKYIWLSRHGESIYNEKKIGGDSSLSERGFQYAKKLEQLVKESEAGEINTLVWTSKMLRTQQTANYLPYKQMKWKLDELDAGVCDGMTYEEIEKEYPEDFKARDNSDKYEYRYRGGESYRDVVIRLEPVIMELERQENVXIITHQAVLRCEYAFMNVPQEEESPWMSIPLHTLIKLEPRAYGTKVTKIKANIPAVSTYKEKGTSSQVGELEQSSTKHLHQLLNDSPLEDKFK
Matching

F26_YEAST Fructose-2,6-bisphosphatase:

MGYSTISNDNDIKVCVIMVGLPARGKSFISQKIIRYLSWLSL
SIKAKCFNVGNYRRDVSGNVPMDAEFFNFENTDNFKL
RELAAQNAIKDIVNFTKEDGSVAVFDATNSTRKRRKWL
LKDICEKNNIQMFLIESWSNDHELIIINNAKDIGSTSPDY
ENSEPHVAEADFLERIRQYERFYEPLEDQKDKDMTFIK
LVNIEEVVINKIRTYLESRIVFYVMNIRPKPYIWLISRHG
ESIYNVEKKIGGDSSLSERGFQYAKKLEQLVKESAGEIN
LTWVTSTLKRRTQQTANYLPYKQLQWKALDELDAGVCD
GMTYEEIEKEYPEDFKARDNDKYEYRYRGGESYRDRVVI
RLEPVIMELERQENVLIITHQAVLRCIYAYFMNVPQEEESP
WMSIPLHTLIKLEPRAYGTKVTKIKAIPAVSTYKEKGTST
QVGELSQSQSTKHLQLLNdSPLIEDKF
OWL Lists: Advantages
OWL Lists: Advantages

- More expressive than rdf:list
OWL Lists: Advantages

› More expressive than rdf:list
  › More flexible and more constrained constructs
e.g., transitivity of isFollowedBy allows statements to be made about (indirectly) following elements
OWL Lists: Advantages

- More expressive than rdf:list
  - More flexible and more constrained constructs
    eg transitivity of isFollowedBy allows statements to be made about (indirectly) following elements
  - Logical statements can be made to represent many different patterns and many different elements
OWL Lists: Advantages

- More expressive than rdf:list
  - More flexible and more constrained constructs
    eg transitivity of isFollowedBy allows statements to be made about (indirectly) following elements
  - Logical statements can be made to represent many different patterns and many different elements
- Knowledge can all be kept in one place (in the ontology)
OWL Lists: Advantages

- More expressive than rdf:list
  - More flexible and more constrained constructs
    - Eg transitivity of isFollowedBy allows statements to be made about (indirectly) following elements
  - Logical statements can be made to represent many different patterns and many different elements
- Knowledge can all be kept in one place (in the ontology)
- Some (very basic) tool support
OWL Lists: Disadvantages
OWL Lists: Disadvantages

- Computationally expensive compared to alternatives (reg exp deterministic finite automata)
OWL Lists: Disadvantages

- Computationally expensive compared to alternatives (reg exp deterministic finite automata)
- Difficult to maintain without specialist tools
OWL Lists: Disadvantages

- Computationally expensive compared to alternatives (reg exp deterministic finite automata)
- Difficult to maintain without specialist tools
  - Current tool support is basic
OWL Lists: Disadvantages

- Computationally expensive compared to alternatives (reg exp deterministic finite automata)
- Difficult to maintain without specialist tools
  - Current tool support is basic
    - Classes, not Individuals (at this point)
OWL Lists: Disadvantages

- Computationally expensive compared to alternatives (reg exp deterministic finite automata)
- Difficult to maintain without specialist tools
  - Current tool support is basic
    - Classes, not Individuals (at this point)
    - Only some of the simple patterns are currently supported
OWL Lists: Disadvantages

- Computationally expensive compared to alternatives (reg exp deterministic finite automata)
- Difficult to maintain without specialist tools
  - Current tool support is basic
    - Classes, not Individuals (at this point)
    - Only some of the simple patterns are currently supported
- Memory intensive (expressions are very large and heavily nested)
OWL Lists: Example (100 elements)
Basic Tools
Basic Tools

- Wizard for basic patterns
Basic Tools

- Wizard for basic patterns
- exact match, starts/ends with, contains
Basic Tools

- Wizard for basic patterns
- exact match, starts/ends with, contains
- Bio-specific for sequences/motifs/fingerprints
Conclusion
Conclusion

‣ Useful for pattern matching with similar expressivity to regular expressions
Conclusion

‣ Useful for pattern matching with similar expressivity to regular expressions

‣ Can use full power of OWL for complex class expressions for sequences and elements
Conclusion

- Useful for pattern matching with similar expressivity to regular expressions
- Can use full power of OWL for complex class expressions for sequences and elements
- Pattern matching surprisingly fast using reasoners
Conclusion

- Useful for pattern matching with similar expressivity to regular expressions
- Can use full power of OWL for complex class expressions for sequences and elements
- Pattern matching surprisingly fast using reasoners
- Tools exist for doing the basics
Conclusion

- Useful for pattern matching with similar expressivity to regular expressions
- Can use full power of OWL for complex class expressions for sequences and elements
- Pattern matching surprisingly fast using reasoners
- Tools exist for doing the basics
- Possible, in theory, to create other structures by reifying the elements
Conclusion

- Useful for pattern matching with similar expressivity to regular expressions
- Can use full power of OWL for complex class expressions for sequences and elements
- Pattern matching surprisingly fast using reasoners
- Tools exist for doing the basics
- Possible, in theory, to create other structures by reifying the elements
  - Trees or Tables?
Basic OWLLists supported by OWL Wizards
http://www.co-ode.org/downloads/wizard/

Demo ontologies at
http://www.co-ode.org/ontologies/lists/

Thanks to Uli Sattler and Bijan Parsia for comments on the hard stuff
EmptyList (The End)

(Thankyou)