There is a real world with real structure. The program of mind has been trained on vast interaction with this world and so contains code that reflects the structure of the world and knows how to exploit it. This code contains representations of real objects in the world and represents the interactions of real objects.

You exploit the structure of the world to make decisions and take actions. Where you draw the line on categories, what constitutes a single object or a single class of objects for you, is determined by the program of your mind, which does the classification. This classification is not random but reflects a compact description of the world, and in particular a description useful for exploiting the structure of the world.

– Eric B. Baum [2004]

Knowledge Graphs and Ontologies

- Is there a flexible way to represent relations?
- How can knowledge/data bases be made to interoperate?

Choosing Individuals and Relations

How to represent: "Pen #5 is red."
suppose the pen is denoted by the constant $pen5$.

Choosing Individuals and Relations

How to represent: "Pen #5 is red."
suppose the pen is denoted by the constant *pen*5.

- $red(pen5)$. It's easy to ask "What's red?"
  Can't ask "what is the color of *pen*5?"

Choosing Individuals and Relations

How to represent: "Pen #5 is red."
suppose the pen is denoted by the constant *pen*5.

- $red(pen5)$. It's easy to ask "What's red?"
  Can't ask "what is the color of *pen*5?"
- $color(pen5, red)$. It's easy to ask "What's red?"
  It's easy to ask "What is the color of *pen*5?"
  Can't ask "What property of *pen*5 has value *red*?"

Choosing Individuals and Relations

How to represent: "Pen #5 is red."
suppose the pen is denoted by the constant *pen*5.

- $red(pen5)$. It's easy to ask "What's red?"
  Can't ask "what is the color of *pen*5?"
- $color(pen5, red)$. It's easy to ask "What's red?"
  It's easy to ask "What is the color of *pen*5?"
  Can't ask "What property of *pen*5 has value *red*?"
- $prop(pen5, color, red)$. It's easy to ask all these questions.

Choosing Individuals and Relations

How to represent: "Pen #5 is red."
suppose the pen is denoted by the constant *pen*5.

- *red*(*pen*5). It's easy to ask "What's red?"
  Can't ask "what is the color of *pen*5?"
- *color*(*pen*5, *red*). It's easy to ask "What's red?"
  It's easy to ask "What is the color of *pen*5?"
  Can't ask "What property of *pen*5 has value *red*?"
- *prop*(*pen*5, *color*, *red*). It's easy to ask all these questions.

prop(*Individual*, *Property*, *Value*) is the only relation needed:
called individual-property-value representation
or triple representation

Universality of *prop*

To represent "a is a parcel"

Universality of *prop*

To represent "a is a parcel"

- $prop(a, type, parcel)$, where *type* is a special property
  Then *parcel* is a class.

Universality of *prop*

To represent "a is a parcel"

- $prop(a, type, parcel)$, where *type* is a special property
  Then *parcel* is a class.
- $prop(a, parcel, true)$, where *parcel* is a Boolean property
  Here parcel is the characteristic function of the class.

Reification

- To represent "Alex gave Chris a book":

Reification

- To represent "Alex gave Chris a book": $gave(alex, chris, b342)$.

Reification

- To represent "Alex gave Chris a book":
  $gave(alex, chris, b342)$.
- Let $b123$ name the booking:

  $prop(ga3545, type, giving\_act)$.

  $prop(ga3545, agent, alex)$.

  $prop(ga3545, patient, b342)$.

  $prop(ga3545, recipient, chris)$.

  $prop(b342, recipient, chris)$.

Reification

- To represent "Alex gave Chris a book":
  $gave(alex, chris, b342)$.
- Let $b123$ name the booking:

  $prop(ga3545, type, giving\_act)$.

  $prop(ga3545, agent, alex)$.

  $prop(ga3545, patient, b342)$.

  $prop(ga3545, recipient, chris)$.

  $prop(b342, recipient, chris)$.
- We have reified the giving action.
- Reify means: to make into an individual.

Reification

- To represent "Alex gave Chris a book":
  $gave(alex, chris, b342)$.
- Let $b123$ name the booking:

  $prop(ga3545, type, giving\_act)$.

  $prop(ga3545, agent, alex)$.

  $prop(ga3545, patient, b342)$.

  $prop(ga3545, recipient, chris)$.

  $prop(b342, recipient, chris)$.
- We have reified the giving action.
- Reify means: to make into an individual.
- What if we want to add the date?

Reification

- To represent "Alex gave Chris a book":
  $gave(alex, chris, b342)$.
- Let $b123$ name the booking:

  $prop(ga3545, type, giving\_act)$.

  $prop(ga3545, agent, alex)$.

  $prop(ga3545, patient, b342)$.

  $prop(ga3545, recipient, chris)$.

  $prop(b342, recipient, chris)$.
- We have reified the giving action.
- Reify means: to make into an individual.
- What if we want to add the date?
- What if we want to add the location?

Knowledge Graphs

When you only have one relation, *prop*, it can be omitted without loss of information.

Logic:

$prop(subject, verb, object)$ or

$rdf(subject, verb, object)$

Knowledge Graphs

When you only have one relation, *prop*, it can be omitted without loss of information.

Logic:

$prop(subject, verb, object)$ or

$rdf(subject, verb, object)$

triple:

$\langle subject, verb, object \rangle$

Knowledge Graphs

When you only have one relation, *prop*, it can be omitted without loss of information.

Logic:

$prop(subject, verb, object)$ or

$rdf(subject, verb, object)$

triple:

$\langle subject, verb, object \rangle$

simple sentence:

individual property value.

subject verb object.

Knowledge Graphs

When you only have one relation, *prop*, it can be omitted without loss of information.

Logic:

$prop(subject, verb, object)$ or

$rdf(subject, verb, object)$

triple:

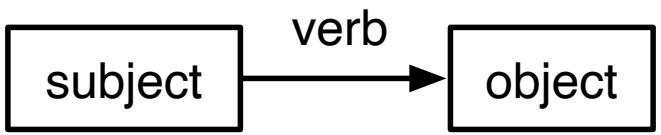
$\langle subject, verb, object \rangle$

simple sentence:

individual property value.

subject verb object.

graphically:



Triples are universal representations of relations

All relations can be represented in terms of triples:

	...	P_j	...
	...	...	...
r_i	...	v_{ij}	...
	...	...	...

can be represented as

Triples are universal representations of relations

All relations can be represented in terms of triples:

	...	P_j	...
	...	...	...
r_i	...	v_{ij}	...
	...	...	...

can be represented as the triple $\langle r_i, P_j, v_{ij} \rangle$.

- r_i is either a primary key or a reified entity.

Triples are universal representations of relations

All relations can be represented in terms of triples:

	...	P_j	...
	...	...	...
r_i	...	v_{ij}	...
	...	...	...

can be represented as the triple $\langle r_i, P_j, v_{ij} \rangle$.

- r_i is either a primary key or a reified entity.
- Examples of reified entities: a booking, a marriage, flight number, transaction number, FIFA Wold Cup Final 2026.

Triples are universal representations of relations

All relations can be represented in terms of triples:

	...	P_j	...
	...	...	...
r_i	...	v_{ij}	...
	...	...	...

can be represented as the triple $\langle r_i, P_j, v_{ij} \rangle$.

- r_i is either a primary key or a reified entity.
- Examples of reified entities: a booking, a marriage, flight number, transaction number, FIFA Wold Cup Final 2026.

$prop(subject, verb, object)$ is the only relation needed: $\langle subject, verb, object \rangle$ triples, semantic network, entity relationship model, knowledge graphs, concept maps, ...

Individuals and Identifiers

- A uniform resource identifier (URI) is a unique name that can be used to identify anything.
- A resource is anything that can be named.

Individuals and Identifiers

- A uniform resource identifier (URI) is a unique name that can be used to identify anything.
- A resource is anything that can be named.
- The modern unicode extension, is internationalized resource identifier (IRI)

Individuals and Identifiers

- A uniform resource identifier (URI) is a unique name that can be used to identify anything.
- A resource is anything that can be named.
- The modern unicode extension, is internationalized resource identifier (IRI)
- A IRI typically has the form of a uniform resource locator (URL), a web address, typically staring with `http://` or `https://`, because URLs are unique.

Individuals and Identifiers

- A uniform resource identifier (URI) is a unique name that can be used to identify anything.
- A resource is anything that can be named.
- The modern unicode extension, is internationalized resource identifier (IRI)
- A IRI typically has the form of a uniform resource locator (URL), a web address, typically staring with `http://` or `https://`, because URLs are unique.
- The IRI denotes the entity, not the web site; if someone uses the IRI they mean the individual denoted by the IRI.

Wikidata IRIs

- Wikidata (`https://www.wikidata.org`) is a free, collaborative knowledge graph with around 1.25 billion triples describing 100 million entities (as of 2022).

Wikidata IRIs

- Wikidata (`https://www.wikidata.org`) is a free, collaborative knowledge graph with around 1.25 billion triples describing 100 million entities (as of 2022).
- The soccer player Christine Sinclair is represented using the identifier "`http://www.wikidata.org/entity/Q262802`"

Wikidata IRIs

- Wikidata (`https://www.wikidata.org`) is a free, collaborative knowledge graph with around 1.25 billion triples describing 100 million entities (as of 2022).
- The soccer player Christine Sinclair is represented using the identifier "`http://www.wikidata.org/entity/Q262802`"
- The identifier "http://schema.org/name" is the property that gives the name of the subject

Wikidata IRIs

- Wikidata (`https://www.wikidata.org`) is a free, collaborative knowledge graph with around 1.25 billion triples describing 100 million entities (as of 2022).
- The soccer player Christine Sinclair is represented using the identifier "`http://www.wikidata.org/entity/Q262802`"
- The identifier "http://schema.org/name" is the property that gives the name of the subject
- "`http://www.wikidata.org/prop/direct/P27`" is the property "country of citizenship".

Wikidata IRIs

- Wikidata (`https://www.wikidata.org`) is a free, collaborative knowledge graph with around 1.25 billion triples describing 100 million entities (as of 2022).
- The soccer player Christine Sinclair is represented using the identifier "`http://www.wikidata.org/entity/Q262802`"
- The identifier "http://schema.org/name" is the property that gives the name of the subject
- "`http://www.wikidata.org/prop/direct/P27`" is the property "country of citizenship".
- Canada is "`http://www.wikidata.org/entity/Q16`"

Wikidata IRIs

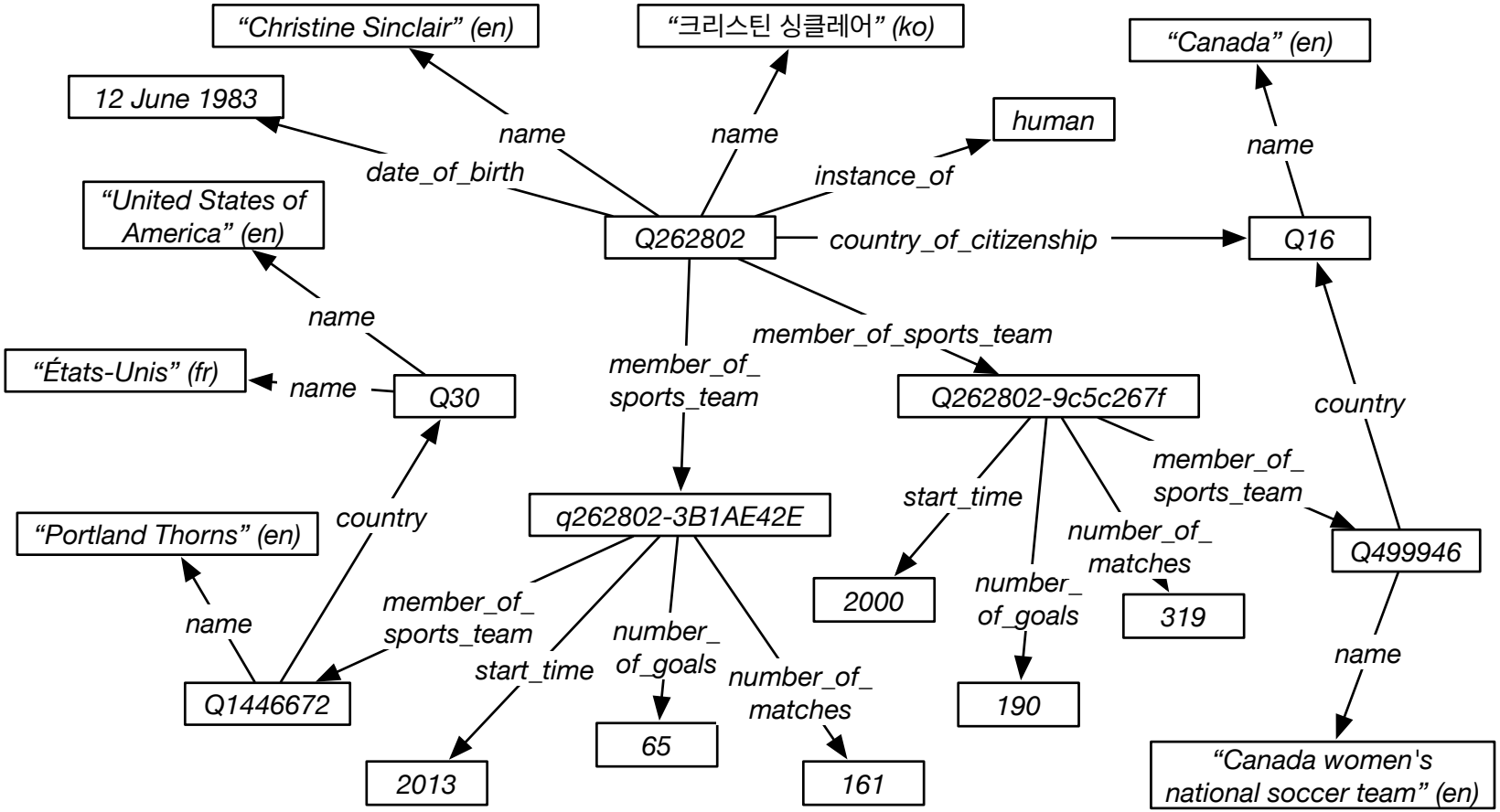
- Wikidata (`https://www.wikidata.org`) is a free, collaborative knowledge graph with around 1.25 billion triples describing 100 million entities (as of 2022).
- The soccer player Christine Sinclair is represented using the identifier "`http://www.wikidata.org/entity/Q262802`"
- The identifier "http://schema.org/name" is the property that gives the name of the subject
- "`http://www.wikidata.org/prop/direct/P27`" is the property "country of citizenship".
- Canada is "`http://www.wikidata.org/entity/Q16`"
- "Christine Sinclair is a citizen of Canada":

Wikidata IRIs

- Wikidata (`https://www.wikidata.org`) is a free, collaborative knowledge graph with around 1.25 billion triples describing 100 million entities (as of 2022).
- The soccer player Christine Sinclair is represented using the identifier "`http://www.wikidata.org/entity/Q262802`"
- The identifier "http://schema.org/name" is the property that gives the name of the subject
- "`http://www.wikidata.org/prop/direct/P27`" is the property "country of citizenship".
- Canada is "`http://www.wikidata.org/entity/Q16`"
- "Christine Sinclair is a citizen of Canada":
  /entity/Q262802 /prop/direct/P27 /entity/Q16

  but all starting with `http://www.wikidata.org`

Part of the Wikidata Knowledge Graph



Accessing Wikidata using Prolog

`https://artint.info/3e/resources/ch16/sem_web.pl`

Warning: naive methods to convert to triples don't work

Projecting onto pairs loses information:

- For example:
  Air Canada flies from New York to Vancouver
  Air Canada flies from Vancouver to Los Angeles

Warning: naive methods to convert to triples don't work

Projecting onto pairs loses information:

- For example:
  Air Canada flies from New York to Vancouver
  Air Canada flies from Vancouver to Los Angeles
- These are true triples:
  $\langle Air\ Canada, Flies\ From, New\ York \rangle$
  $\langle Air\ Canada, Flies\ To, Los\ Angeles \rangle$

Warning: naive methods to convert to triples don't work

Projecting onto pairs loses information:

- For example:
  Air Canada flies from New York to Vancouver
  Air Canada flies from Vancouver to Los Angeles
- These are true triples:
  $\langle Air\ Canada, Flies\ From, New\ York\rangle$
  $\langle Air\ Canada, Flies\ To, Los\ Angeles\rangle$
- However, Air Canada does not fly from New York to Los Angeles.
  The information about flights is lost!