Feature Extraction with Description Logics Functional Subsumption

Rodrigo de Salvo Braz
Dan Roth

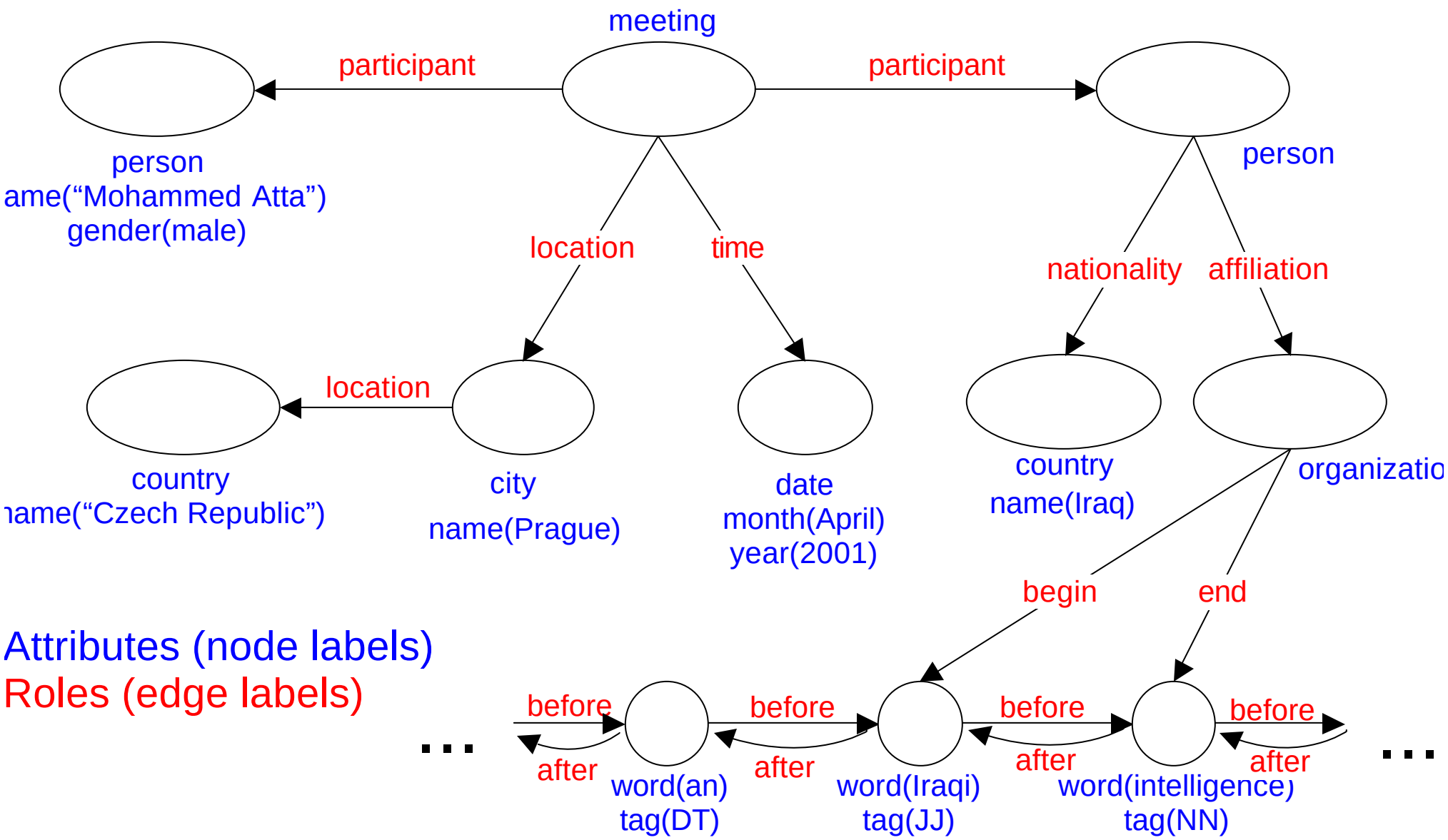
University of Illinois at Urbana-Champaign

A conflict

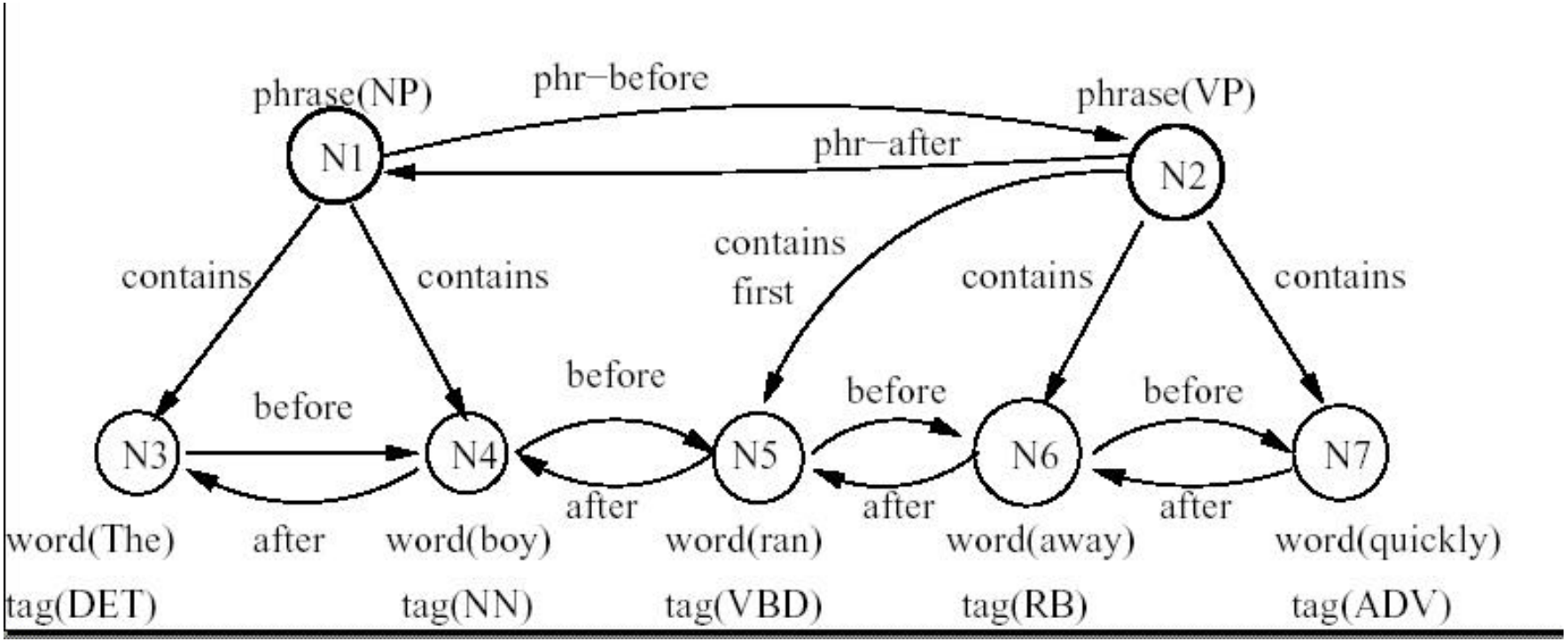
- Most machine learning algorithms use feature vectors as inputs.
- Most data is best represented as structured data.
- Feature extraction is the conversion from one to the other (and may be most of the work).

Structured data – I

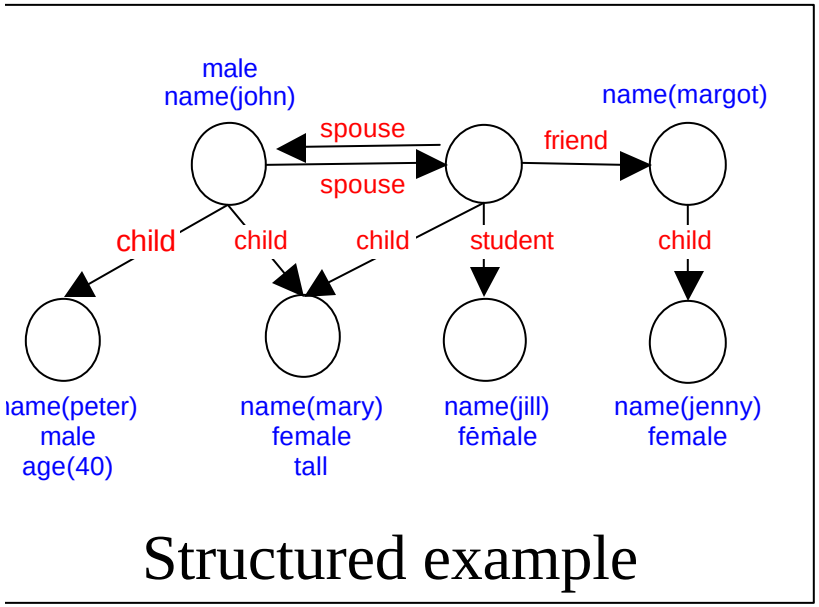
Mohammed Atta met with an Iraqi intelligence agent in Prague in April 2001



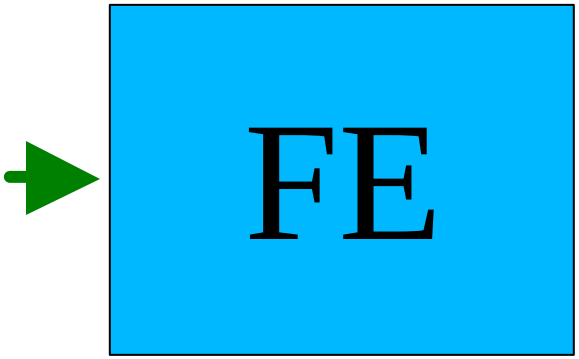
Structured data – II

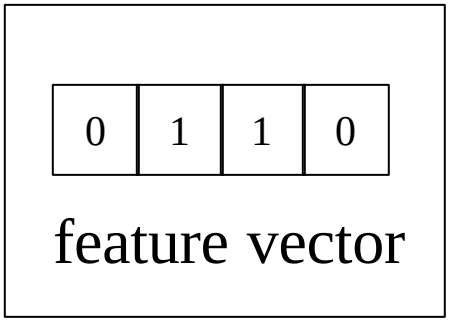


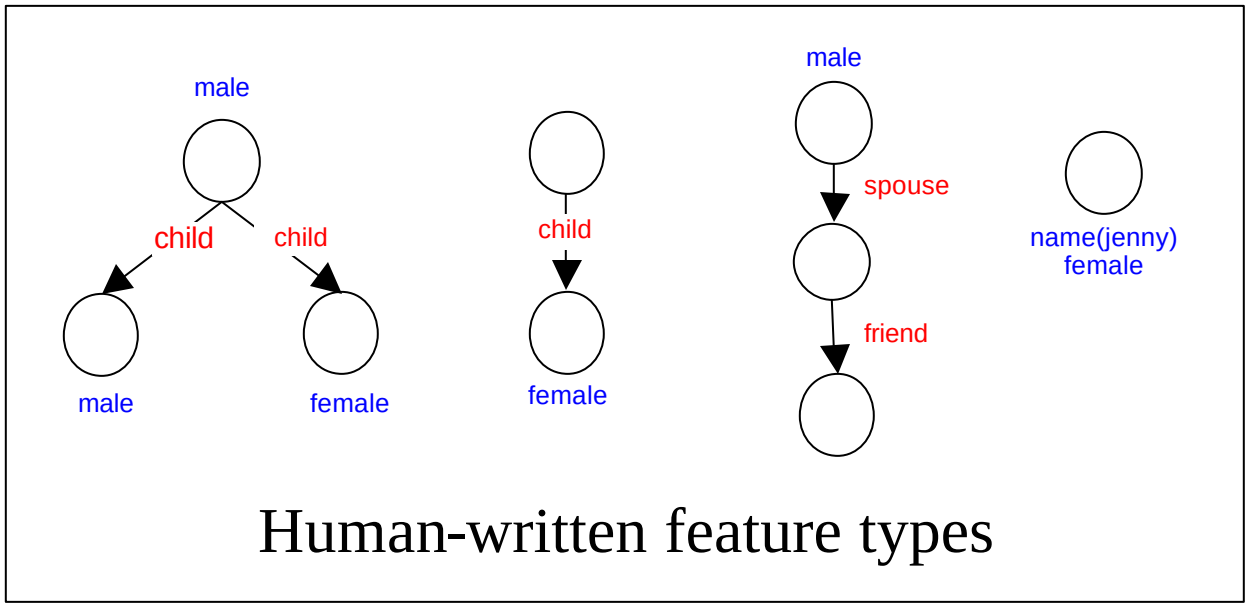
Feature Extraction



Structured example







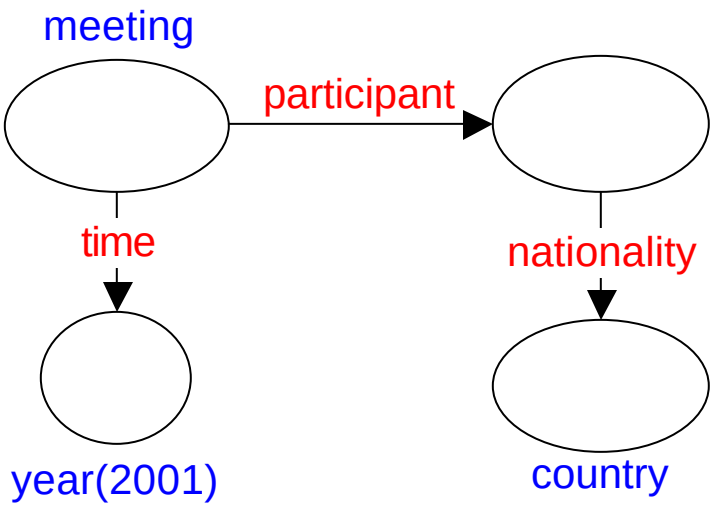
Human-written feature types

Feature Extraction

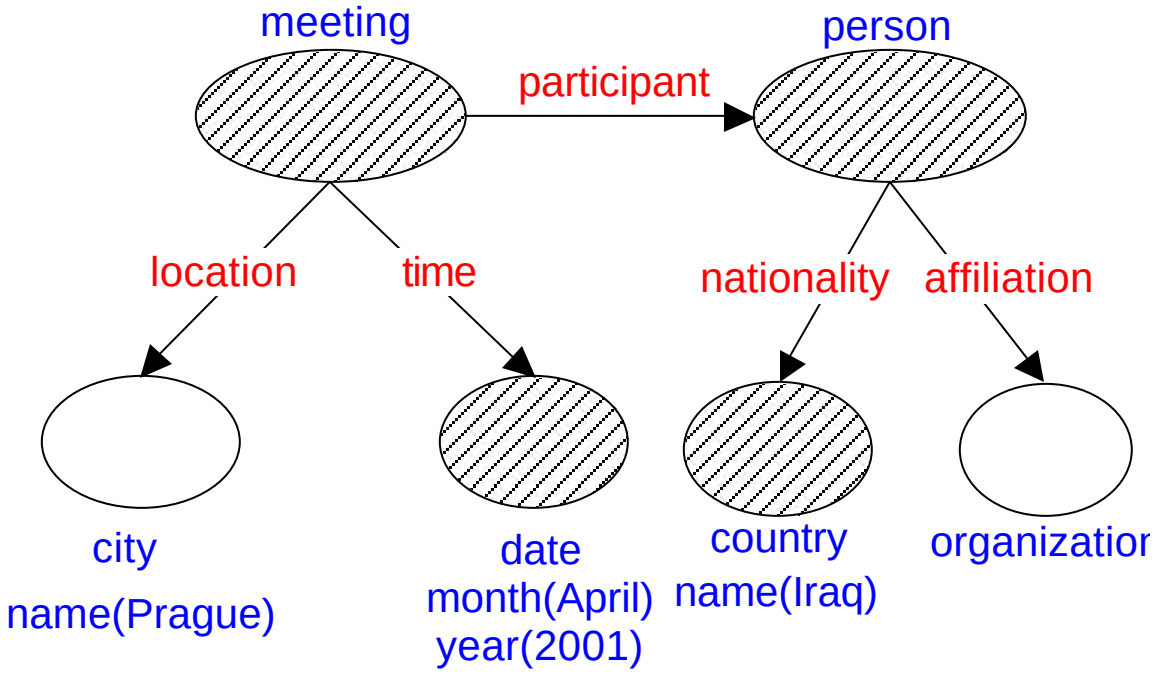
- *Typically done in ad hoc fashion:*
  - *Prevents general analysis;*
  - *Prevents Feature Extraction/Learning unified analysis (e.g. kernels).*
- *Using a language is tricky*
  - *Type of inference.*
  - *May be intractable if not careful.*

A language for declaring which features to generate

Feature type specifications by directed trees

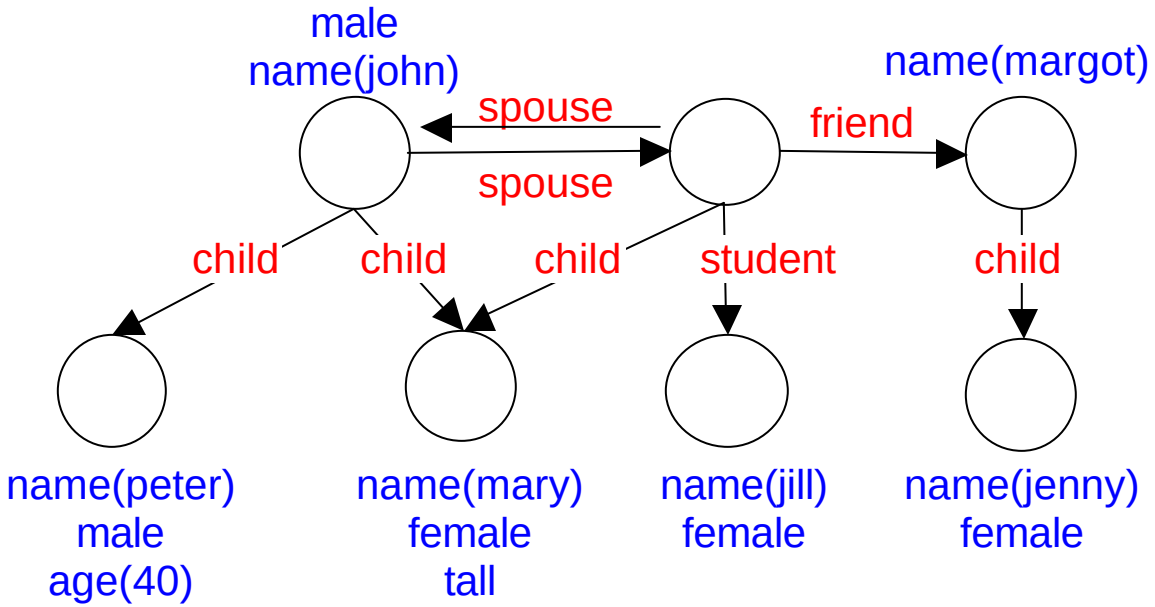


Example segment

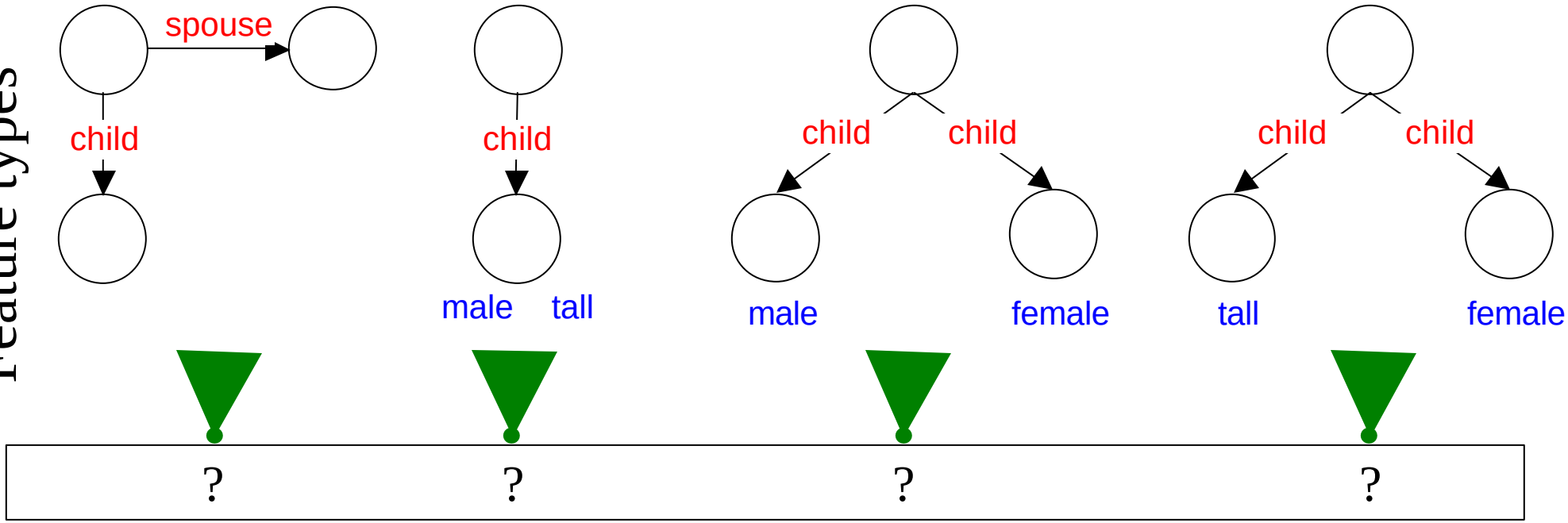


Generating feature vectors

Example

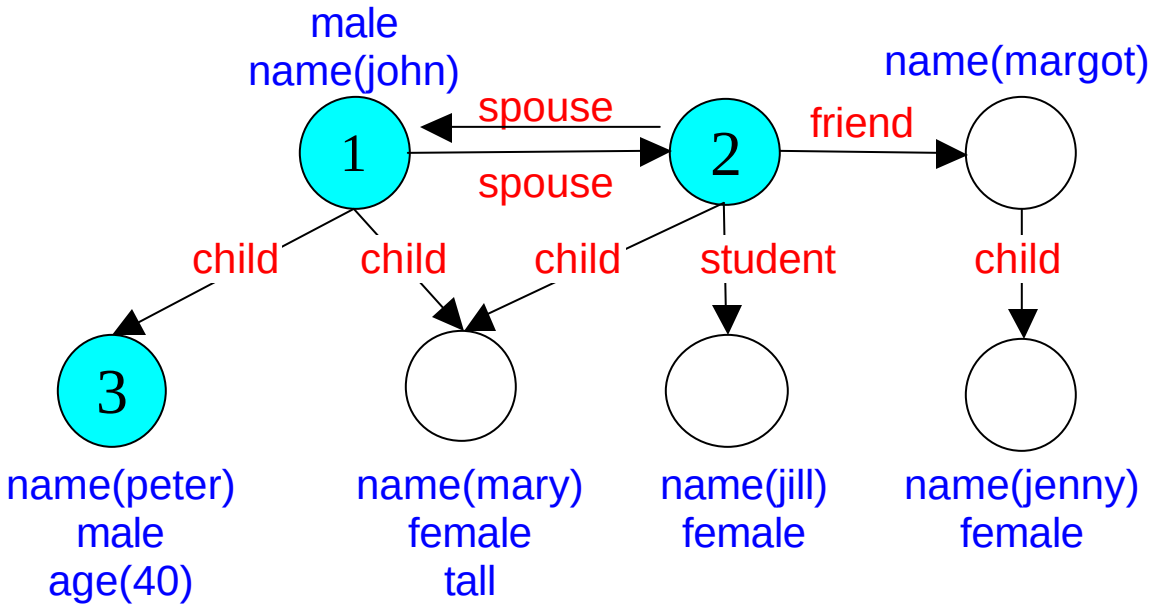


Feature types

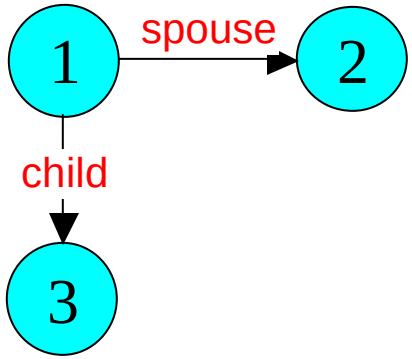


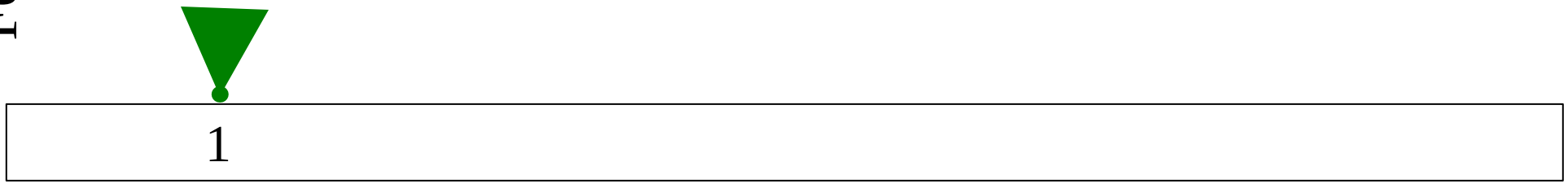
Generating feature vectors

Example



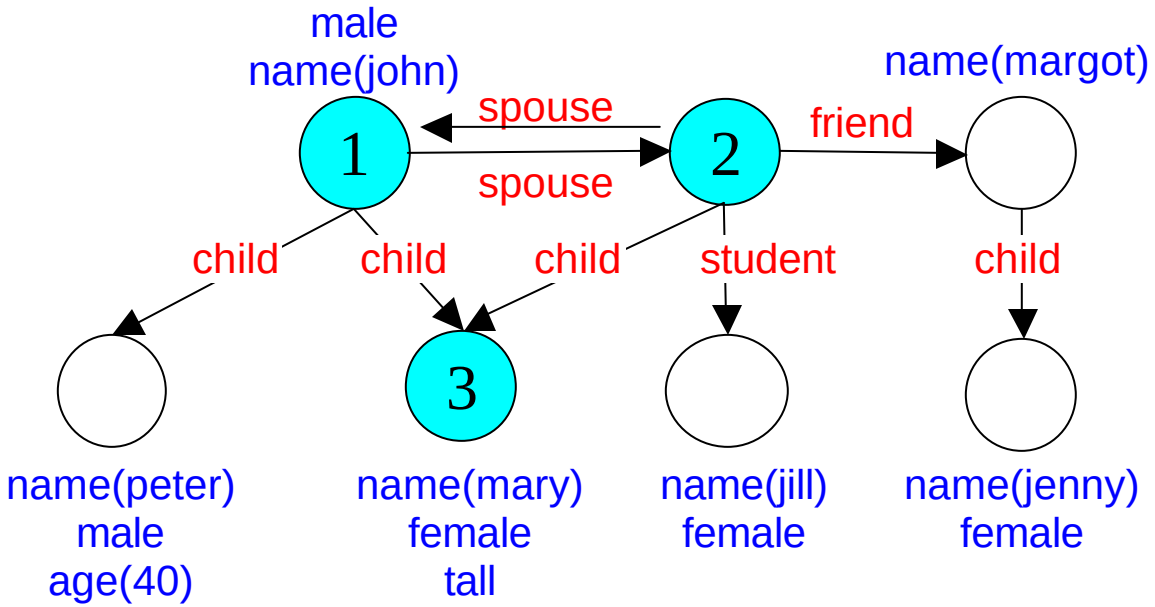
Feature types



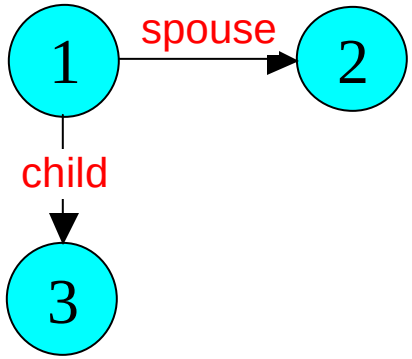


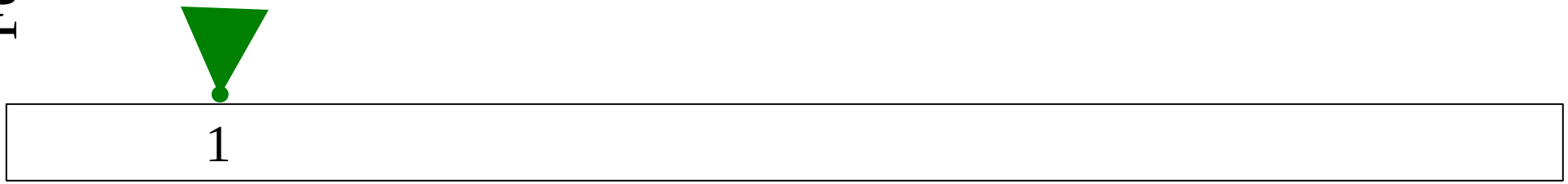
Generating feature vectors

Example



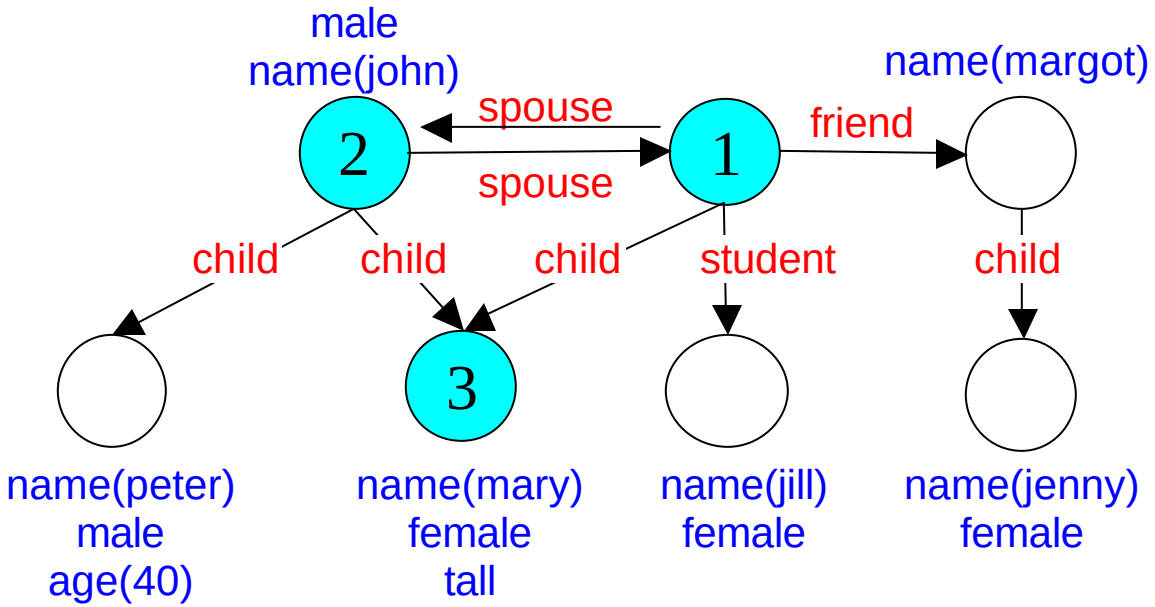
Feature types



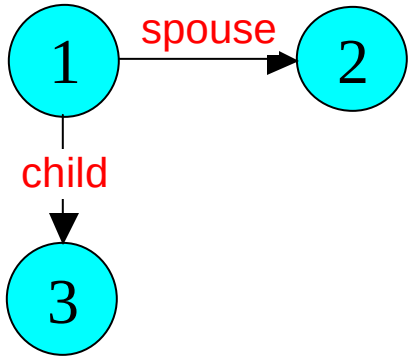


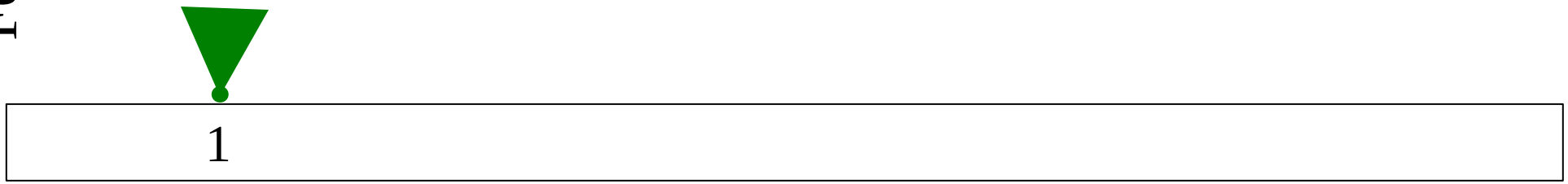
Generating feature vectors

Example



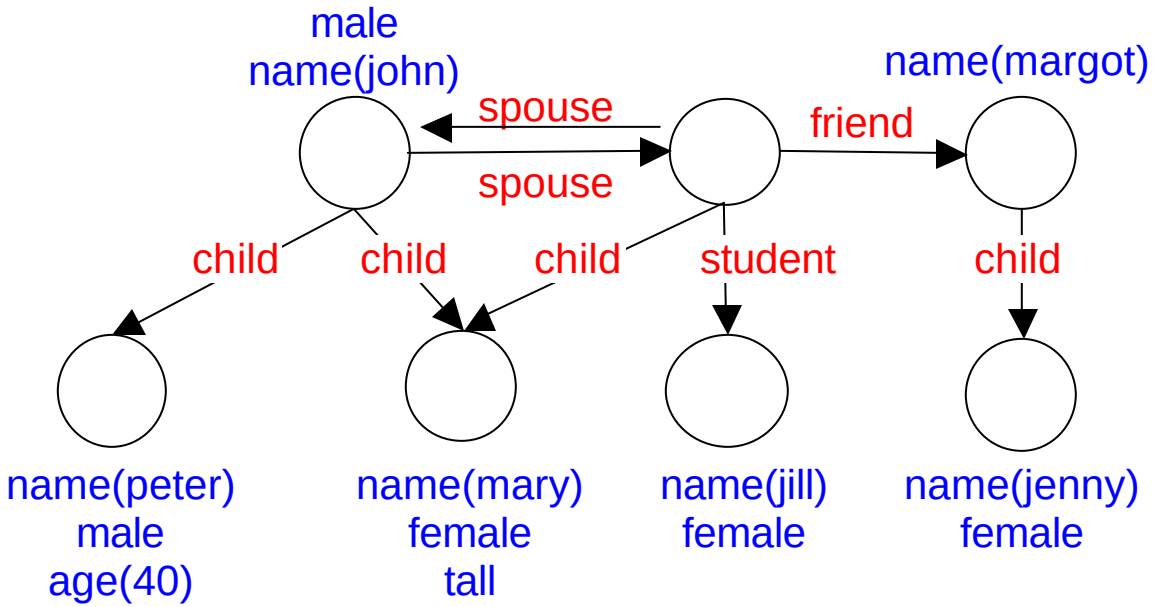
Feature types



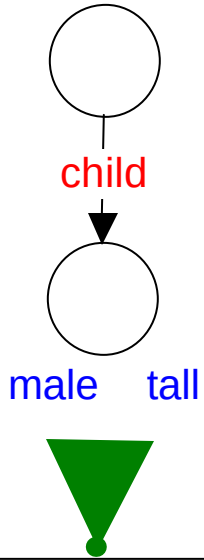


Generating feature vectors

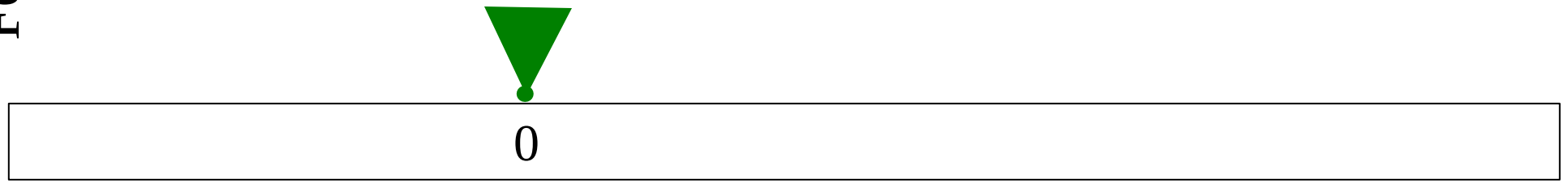
Example



Feature types

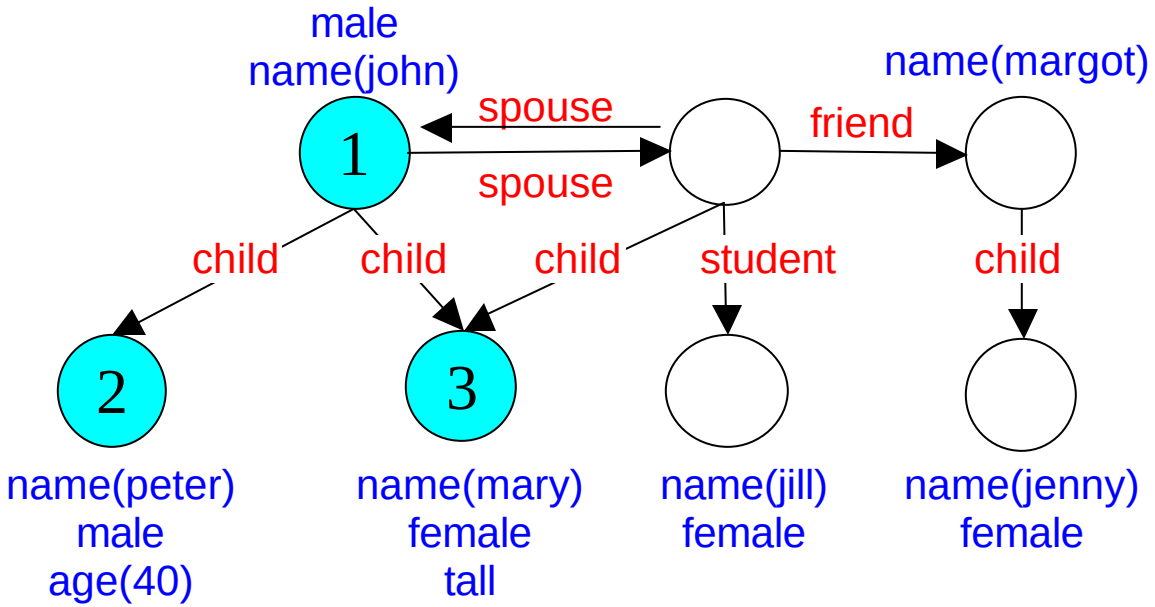


Nothing like this
in the example!

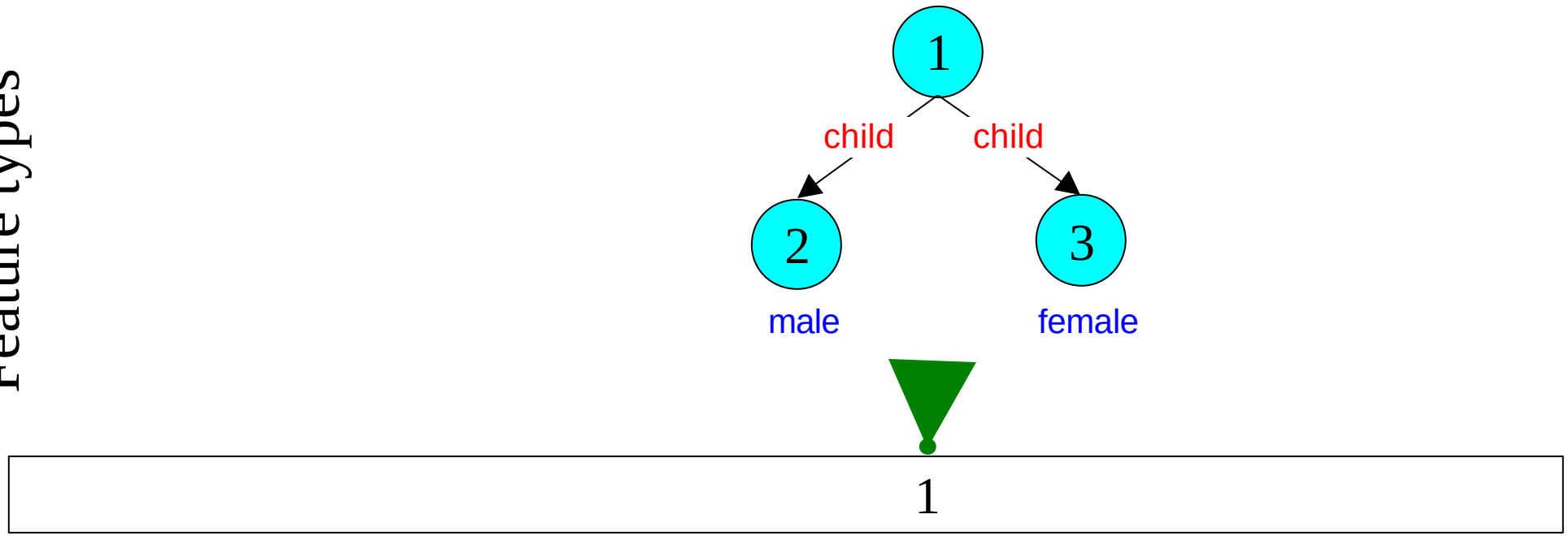


Generating feature vectors

Example

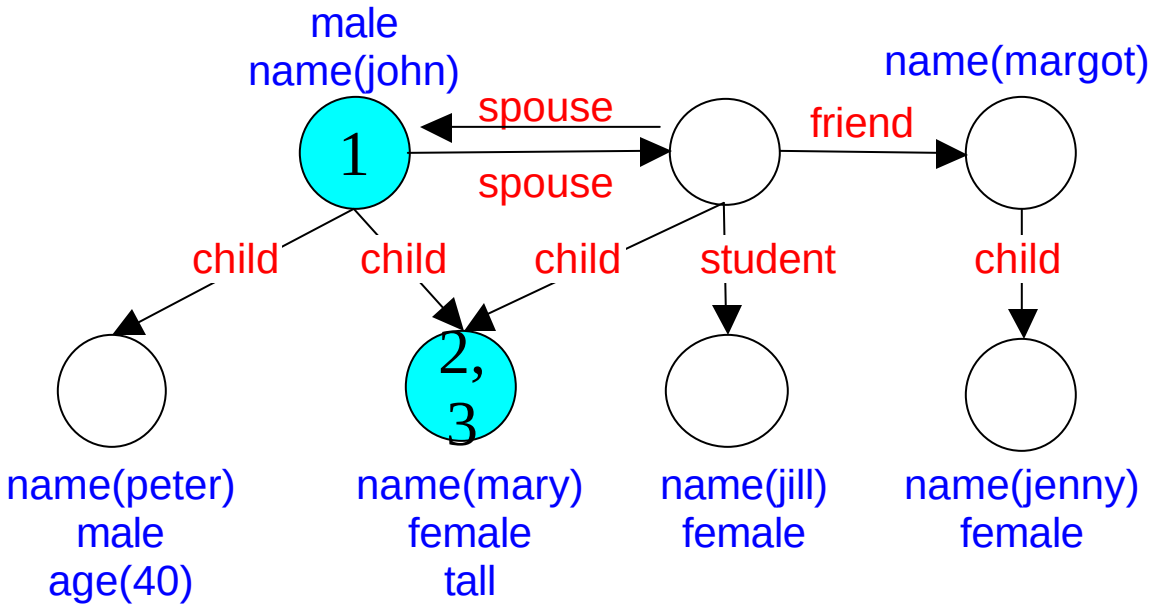


Feature types

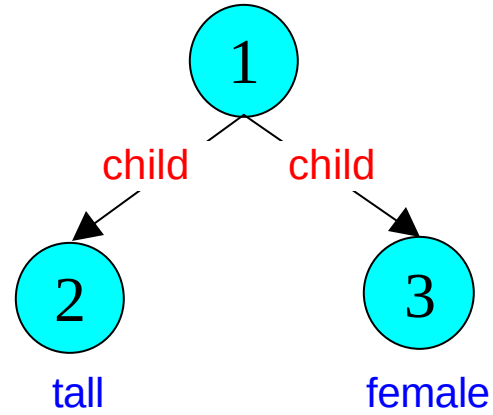


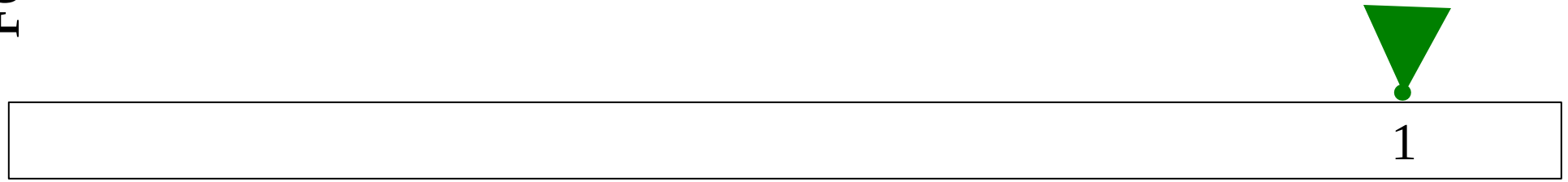
Generating feature vectors

Example



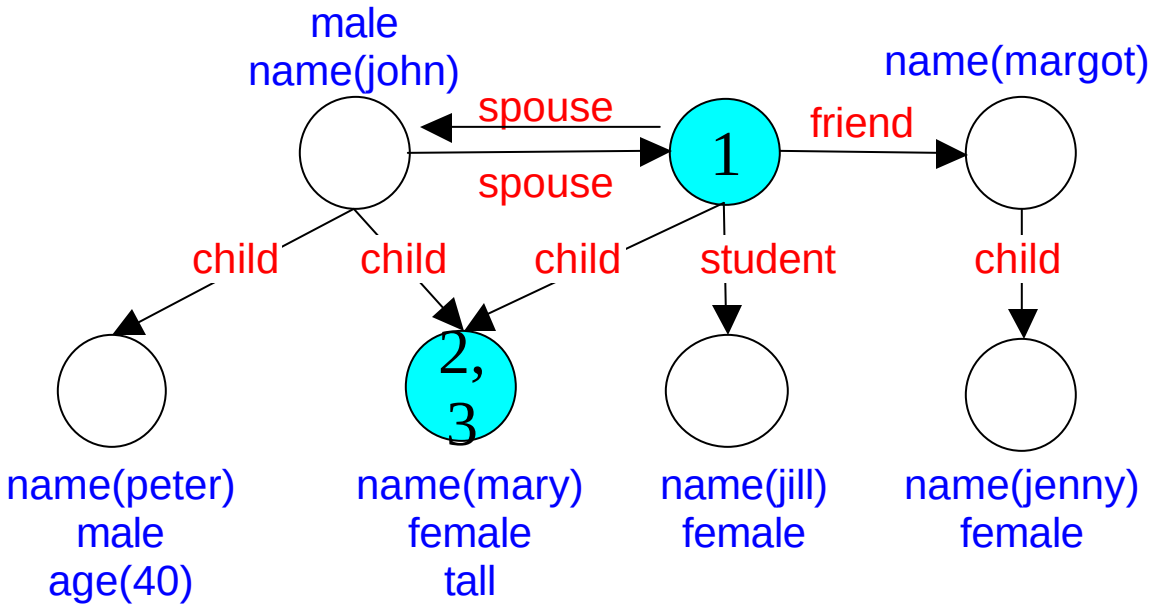
Feature types



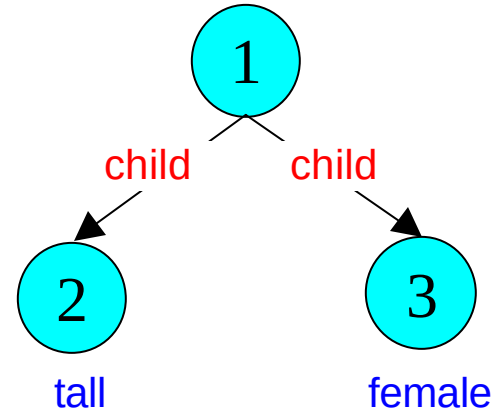


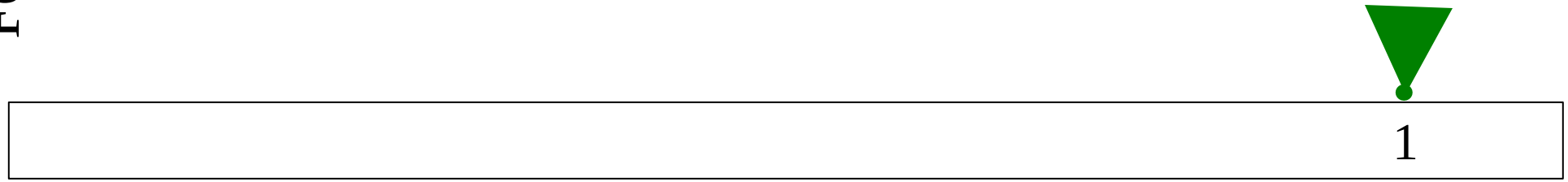
Generating feature vectors

Example



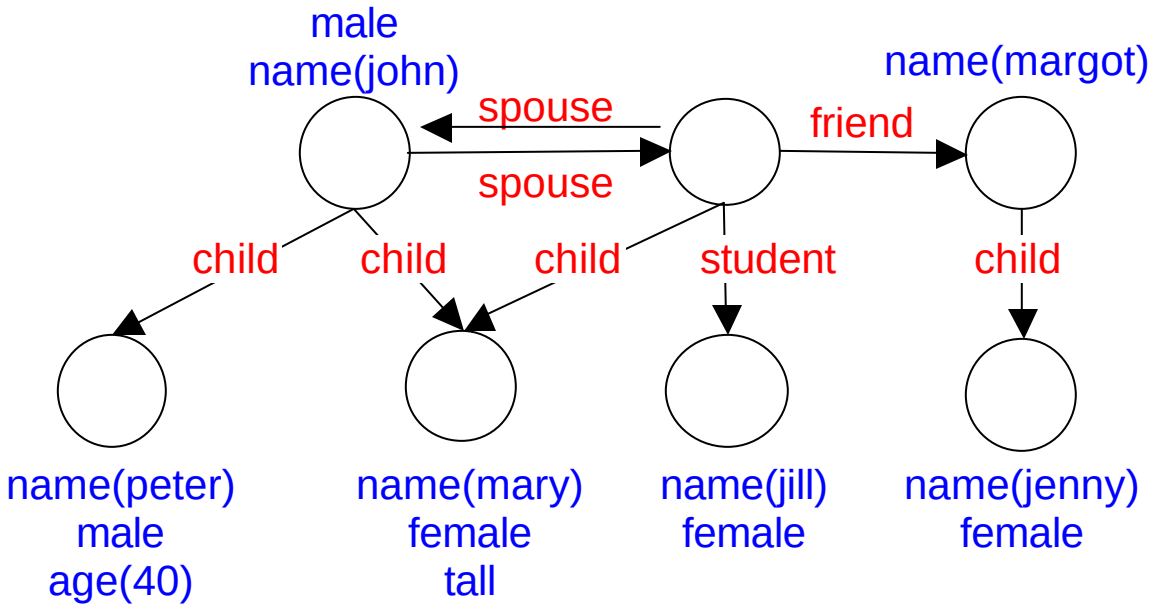
Feature types



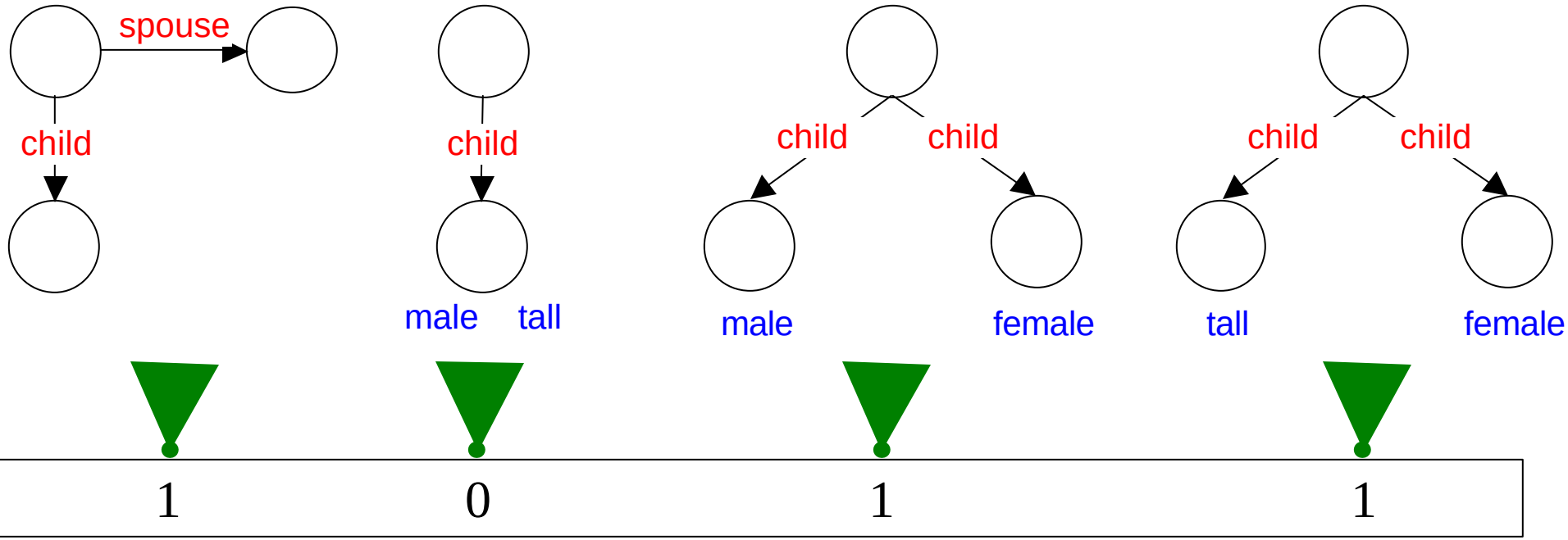


Generating feature vectors

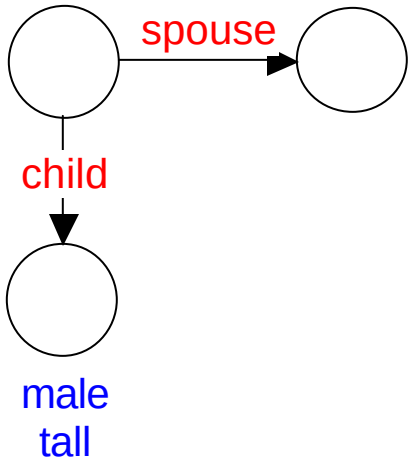
Example



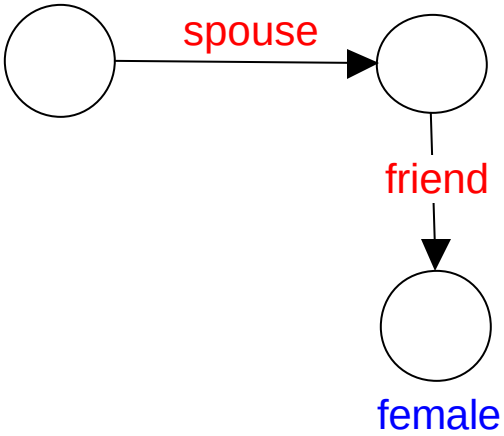
Feature types



Feature Description Logics



(AND
(SOME spouse ANY)
(SOME child (AND male tall)))

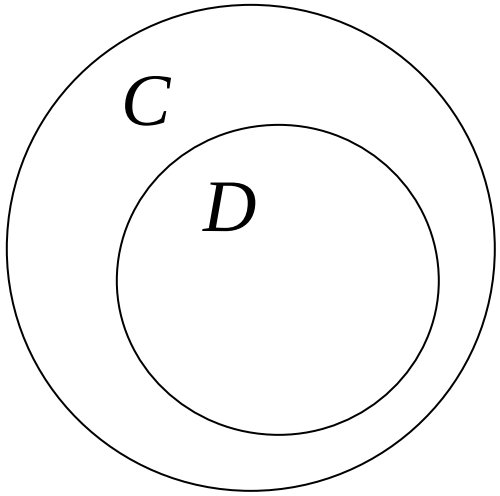


(SOME spouse
(SOME friend female))

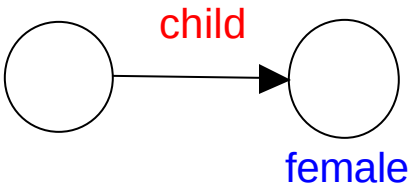
Subsumption

- A description C *subsumes* (⊇) a description D if every individual in *D* must be in *C*, no matter the interpretation.
- Subsumption is tractable.

C = (AND
(SOME spouse ANY)
(SOME child male))
D = (AND
(SOME spouse (SOME student ANY))
(SOME child (AND tall male))
(SOME child female))

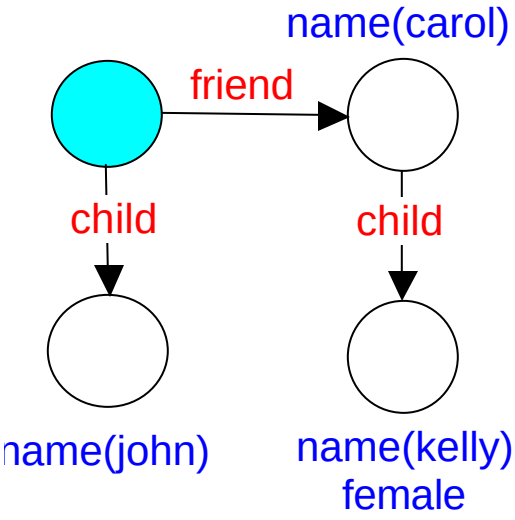


Feature extraction as subsumption

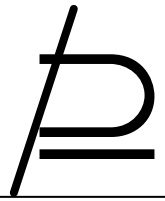


Feature type

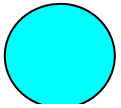
Example



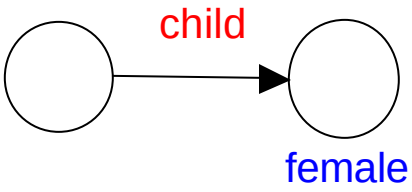
(SOME child female)



(AND
  SOME friend
    (AND
      name(carol)
      SOME child (AND name(kelly) female))
  SOME child name(john))

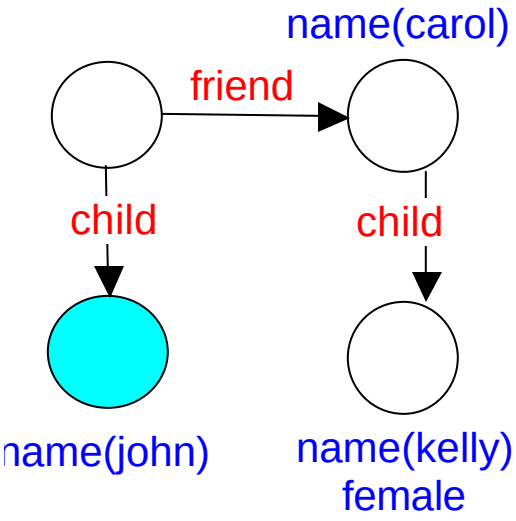
Description of node 

Feature extraction as subsumption



Feature type

Example



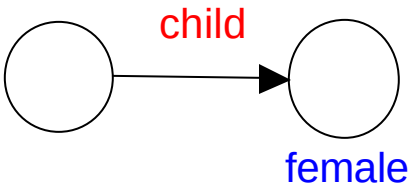
(SOME child female)

$\not\trianglerighteq$

name(john)

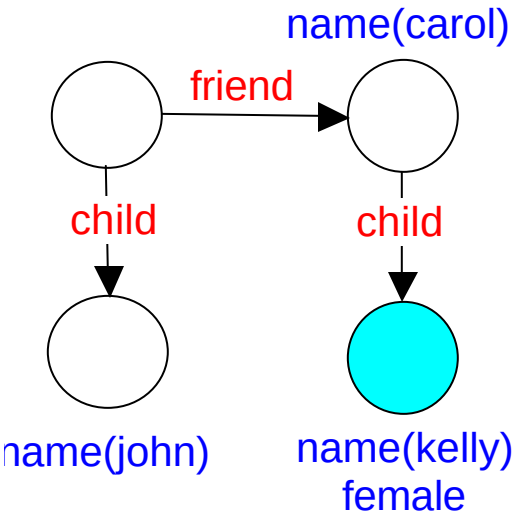
Description of node

Feature extraction as subsumption



Feature type

Example



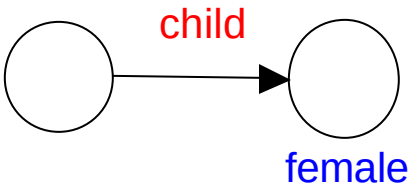
(SOME child female)

$\not\supseteq$

(AND name(kelly) female)

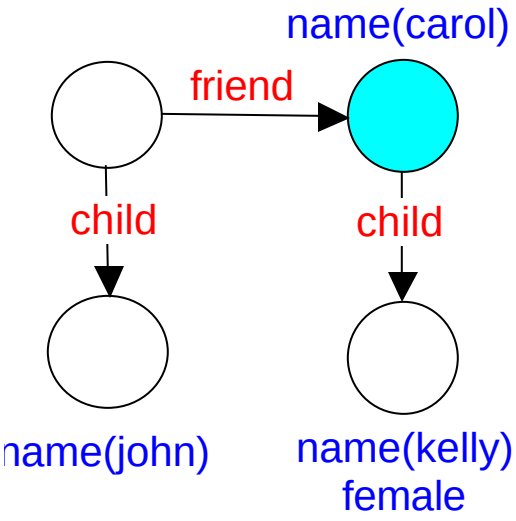
Description of node

Feature extraction as subsumption

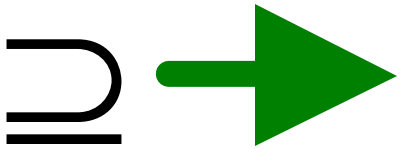


Feature type

Example

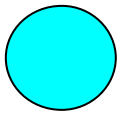


(SOME child female)

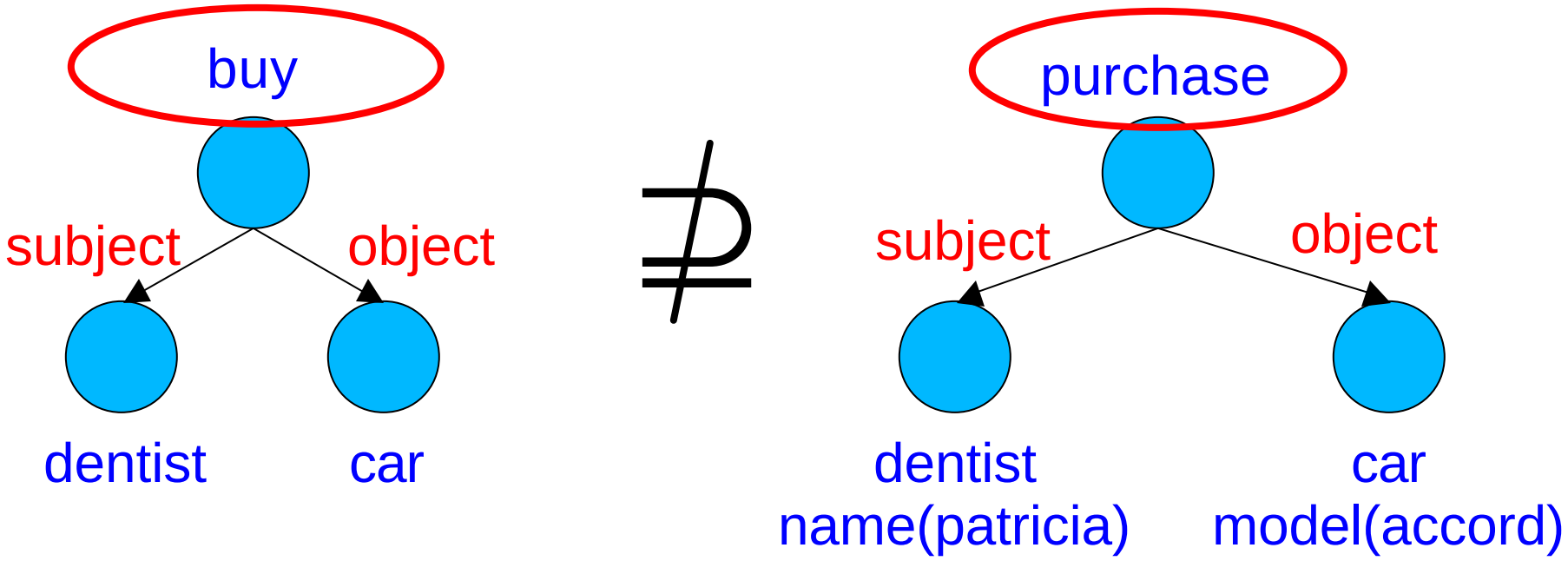


active feature!

(AND
  name(carol)
  SOME child (AND name(kelly) female))

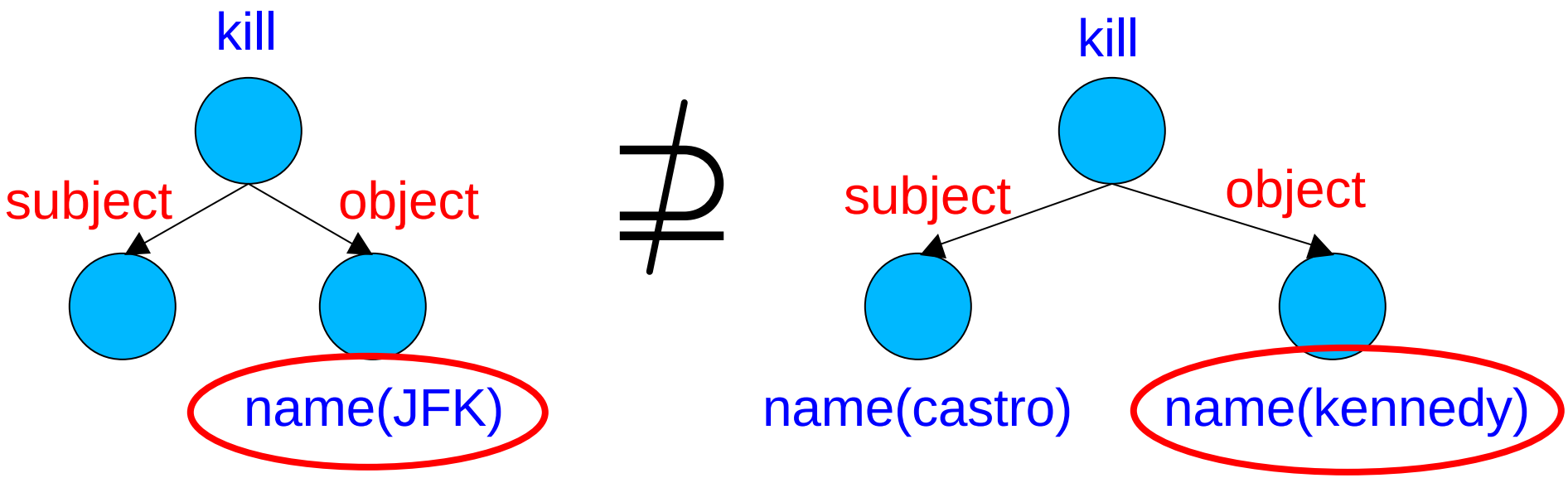
Description of node 

A problem in practice

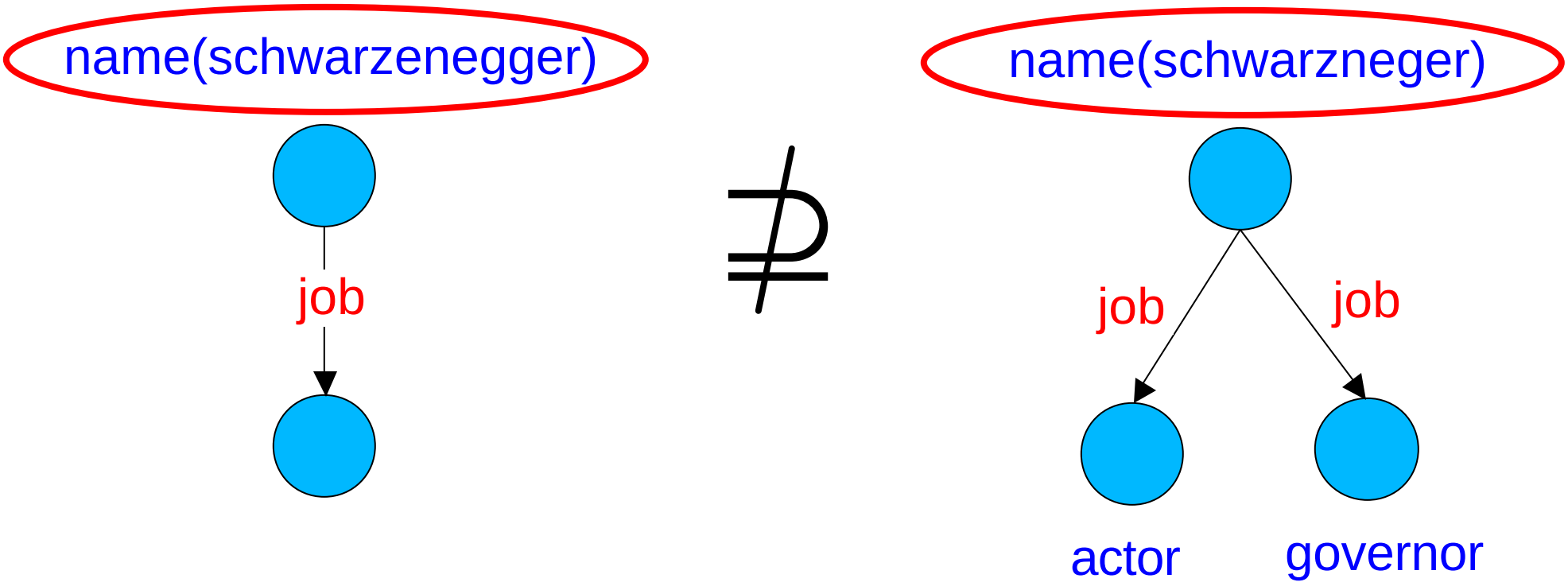


Subsumption would be natural in this case but does not occur

A problem in practice



A problem in practice



Make comparison more flexible

- At core of subsumption algorithm is the comparison of attributes:

... if (attr1 == attr2) ...

- We simply make that a function call:

... if (f (attr1, attr2) == 1) ...

Is this just a hack?

What about the nice DL semantics?

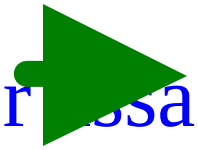
Is this just a hack?

*W*hat about the nice DL semantics?
n fact, equivalent to “shallow OR” (*tractable*).

Is this just a hack?

What about the nice DL semantics?
In fact, equivalent to “shallow OR” (*tractable*).
Replace any attr by (OR a_1 a_2 ... a_n)
where f(attr, a_i) = 1.

AND kill
(SOME object JFK))



AND (OR kill murder assassinate)
(SOME object (OR
JFK kennedy “John F. Kennedy” ...)))

Why not just use shallow OR then?

- Function is an implicit representation.
- We may incorporate procedural knowledge:
  - Typos;
  - Similar sounding words;
  - Context-sensitive knowledge.

Take home message

- Feature Description Logics provides an expressive way to deal with structured examples.
- Syntax choices render it tractable.
- Allows for FE-learning integrated approaches like kernels (Cumby & Roth 2003).
- Can be made even more expressive with little extra cost by functional subsumption.

The End