Ontologies and Reasoning

Pedro Szekely
University of Southern California

computers are dimwits

name	rank
Bob	Captain
Sue	Sargent
Mary	Admiral
Joe	Sargent

```
select ?x  { ?x  rank  "Sargent" }

select ?x  { ?x  a  Person }

select ?x  { ?x  a  MilitaryPerson }
```

inferencing

inferencing

think RDF triples

inferencing

I tell the computer some triples

it infers more triples

name	rank
Bob	Captain
Sue	Sargent
Mary	Admiral
Joe	Sargent

bob
a foaf:Person;
foaf:name “Bob”;
ex:rank “Captain” .

sue
a foaf:Person;
foaf:name “Sue”;
ex:rank “Sargent” .

select ?x { ?x rank “Sargent” }

select ?x { ?x a Person }

select ?x { ?x a MilitaryPerson }

name	rank
Bob	Captain
Sue	Sargent
Mary	Admiral
Joe	Sargent

bob
a foaf:Person;
foaf:name “Bob”;
ex:rank “Captain” .

sue
a foaf:Person;
foaf:name “Sue”;
ex:rank “Sargent” .

select ?x { ?x a MilitaryPerson }

what inference do I need?

name	rank
Bob	Captain
Sue	Sargent
Mary	Admiral
Joe	Sargent

bob a foaf:Person;
foaf:name “Bob”;
ex:rank “Captain” .

sue a foaf:Person;
foaf:name “Sue”;
ex:rank “Sargent” .

select ?x { ?x a MilitaryPerson }

ONE SMALL **RDFSCHEMA** STATEMENT

ex:rank rdfs:domain ex:MilitaryPerson .

RDF Schema Vocabulary

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

Classes

rdfs:Resource
rdfs:Class
rdfs:Literal
rdfs:Datatype
rdf:XMLLiteral
rdf:Property

Properties

rdfs:range
rdfs:domain
rdf:type
rdfs:subClassOf
rdfs:subPropertyOf
rdfs:label
rdfs:comment

Utility Properties

rdfs:seeAlso
rdfs:isDefinedBy
rdf:value

Limitations of RDFS

RDF Schema is a simple modelling language:

- Cannot define: Boolean combinations of classes, disjointness, cardinality restrictions, ..
- Cannot define properties of properties: unique, inverse, transitive, …
- No mechanism of specifying necessary and sufficient conditions for class membership.
  - Ex: If John has a vehicle which is 7ft high, has wide wheels and loading space is 3 m3, then we should be able to reason that John has a SUV,
    if the necessary and sufficient conditions for a vehicle being an SUV are height > 6ft & wide wheels & loading space > 2 m3

Therefore we need an ontology layer on top of RDF and RDF Schema → OWL

OWL vs RDFS

Ideally, OWL would extend RDF Schema to be consistent with the layered architecture of the Semantic Web

However, directly combining RDF Schema with classical logic leads to problems:

- RDF allows resources to be instances and classes (elements and sets) at the same time, which leads to paradoxes
- RDF/RDFS was formalized using non-standard semantics:
  - RDF Semantics: http://www.w3.org/TR/rdf-mt/

inferencing

RDFSchema OWL

Basic Idea

I state a few OWL axioms

I load lots of triples

the system infers lots of new triples

OWL Building Blocks

Classes

Properties

Individuals

Unique Name Assumption

```
bob   a foaf:Person;
      foaf:name "Bob";
      ex:rank "Captain" .
```

```
sue   a foaf:Person;
      foaf:name "Sue";
      ex:rank "Sargent" .
```

Unique Name Assumption = TRUE:

bob and sue refer to different individuals in the world

Unique Name Assumption

bob a foaf:Person;
foaf:name “Bob”;
ex:rank “Captain” .

sue a foaf:Person;
foaf:name “Sue”;
ex:rank “Sargent” .

Unique Name Assumption = TRUE:

bob and sue refer to different individuals in the world

Unique Name Assumption = FALSE:

bob and sue may refer to the same individual in the world

OWL

Unique Name Assumption = FALSE

Open World, Closed World

I tell the computer some triples

Open World, Closed World

I tell the computer some triples

Closed World:

Open World, Closed World

I tell the computer some triples

Closed World:
if I don't tell it something, assume it's false

Open World, Closed World

I tell the computer some triples

Closed World:
if I don't tell it something, assume it's false

Open World:

Open World, Closed World

I tell the computer some triples

Closed World:
if I don't tell it something, assume it's false

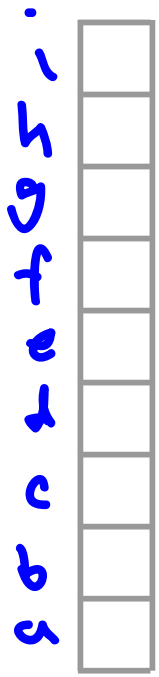
Open World:
if I don't tell it something, don't assume anything
I might tell it later that it is true

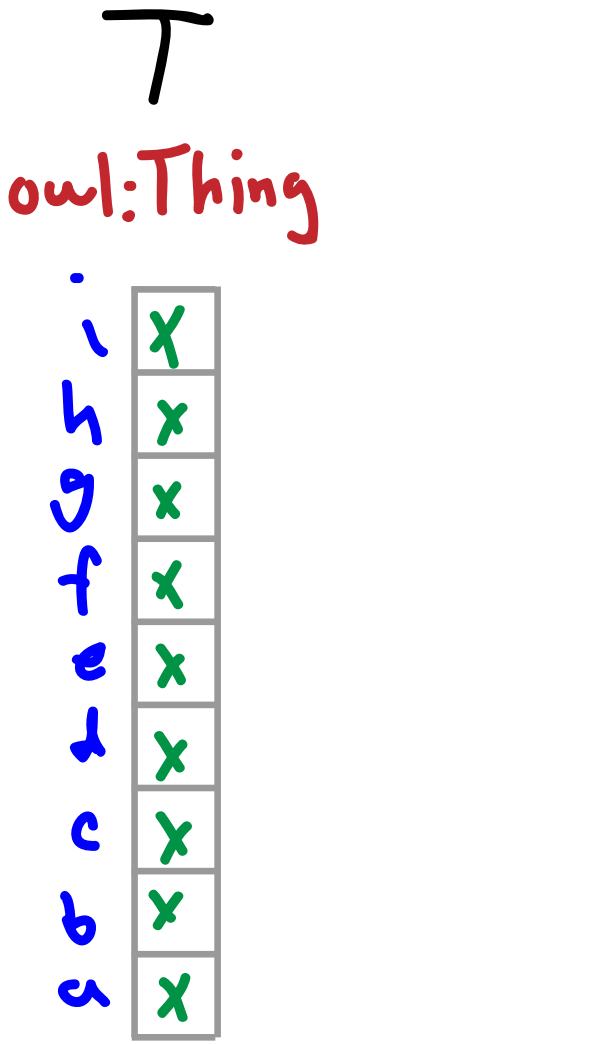
OWL

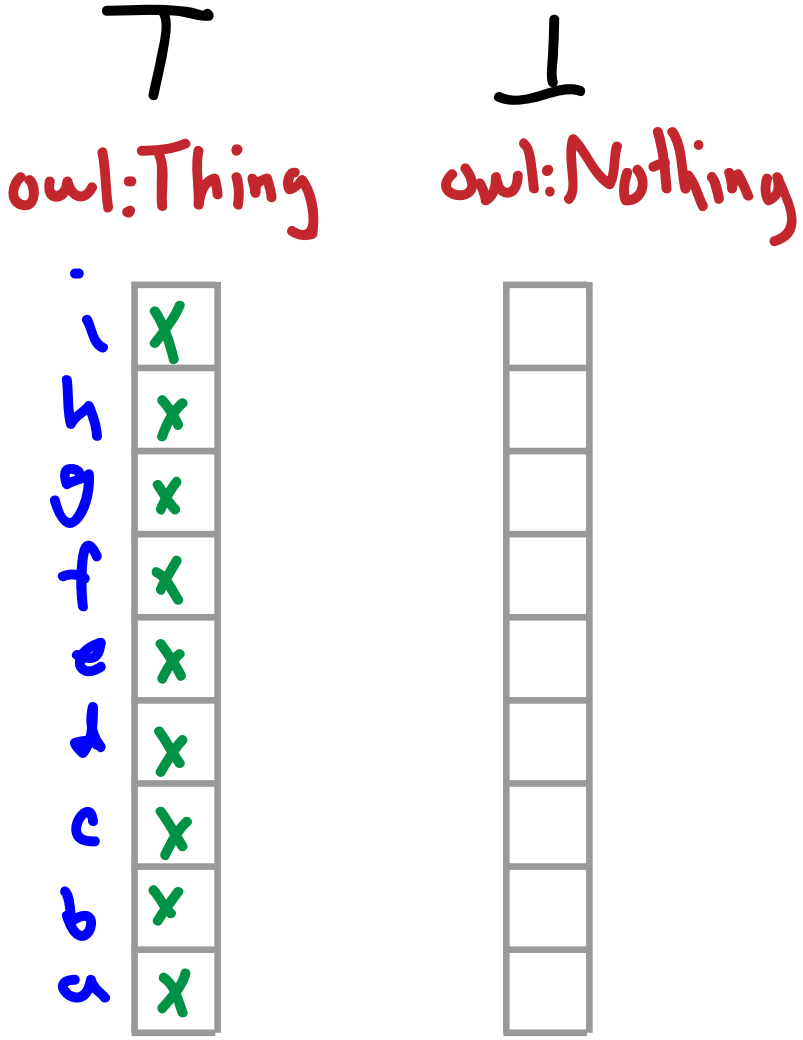
Open world or closed world = OPEN

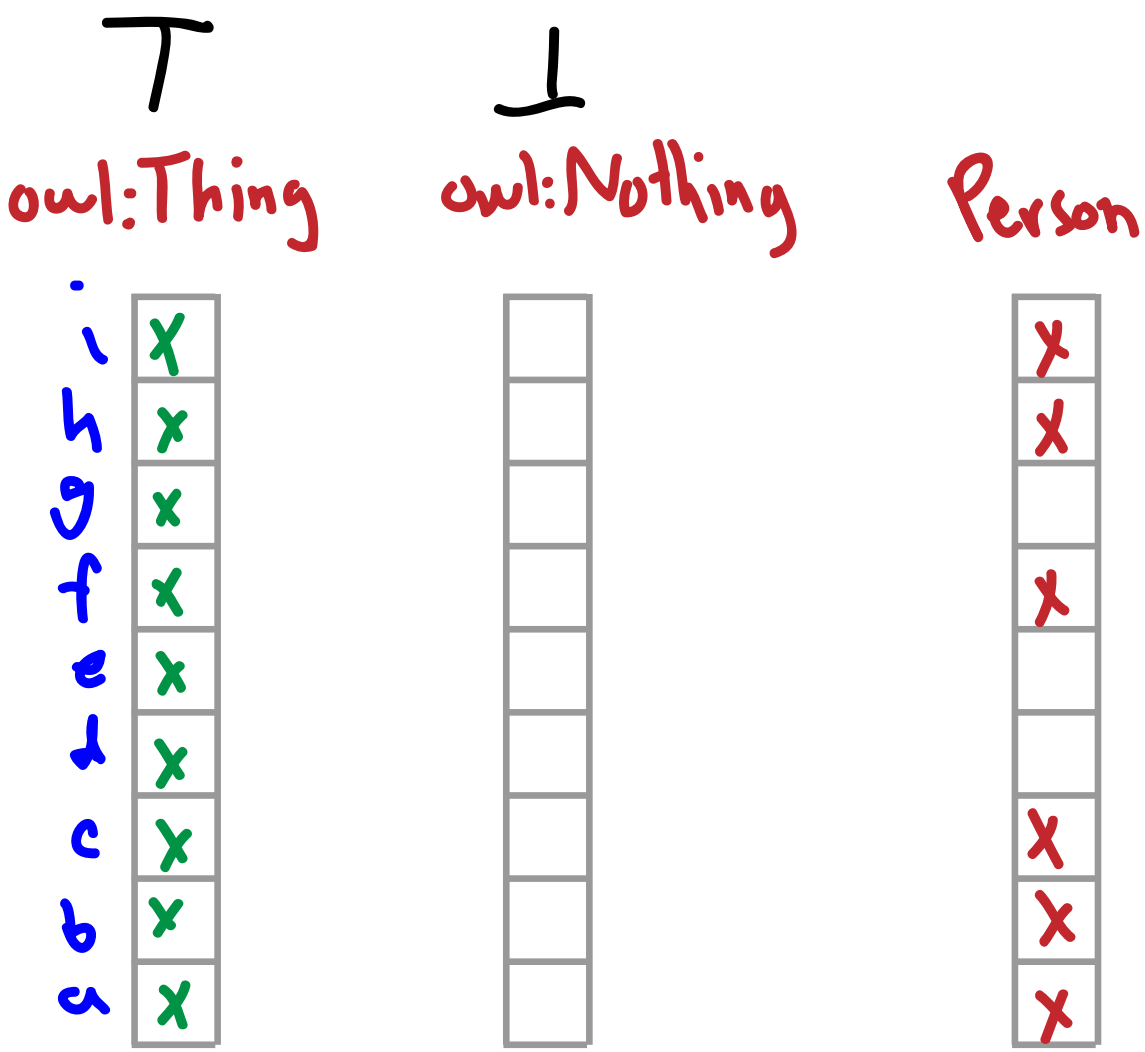
Classes

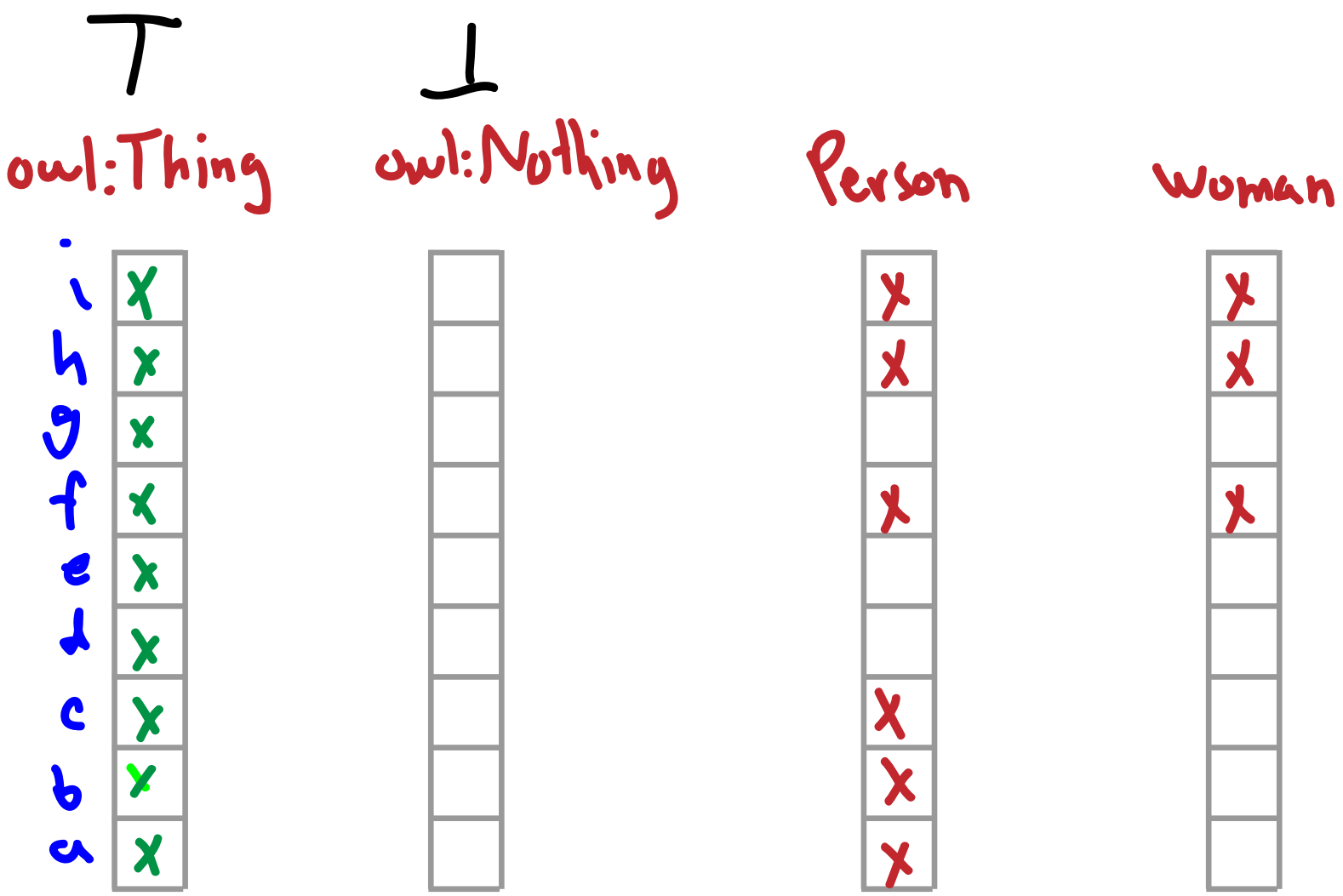
all individuals

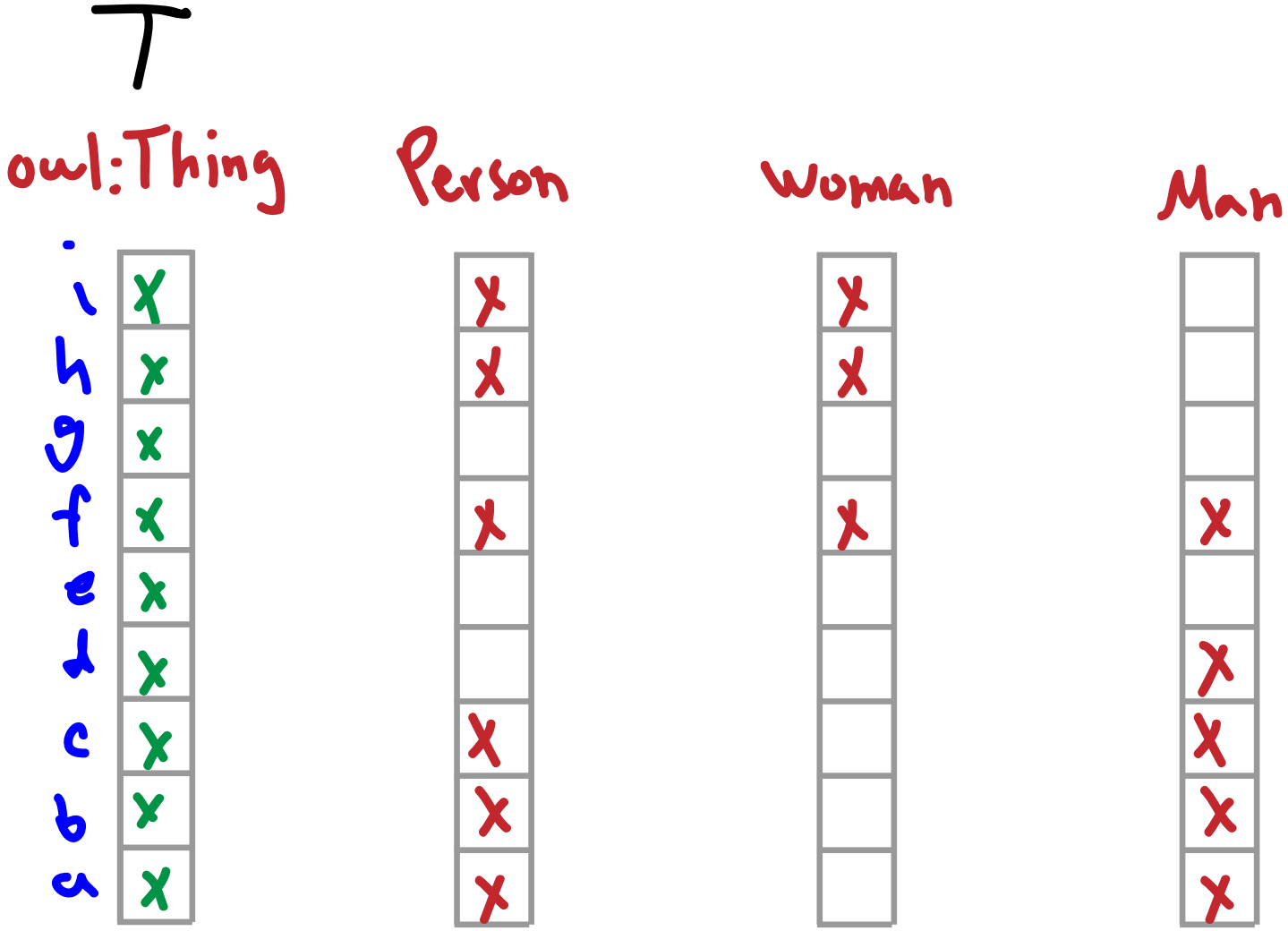


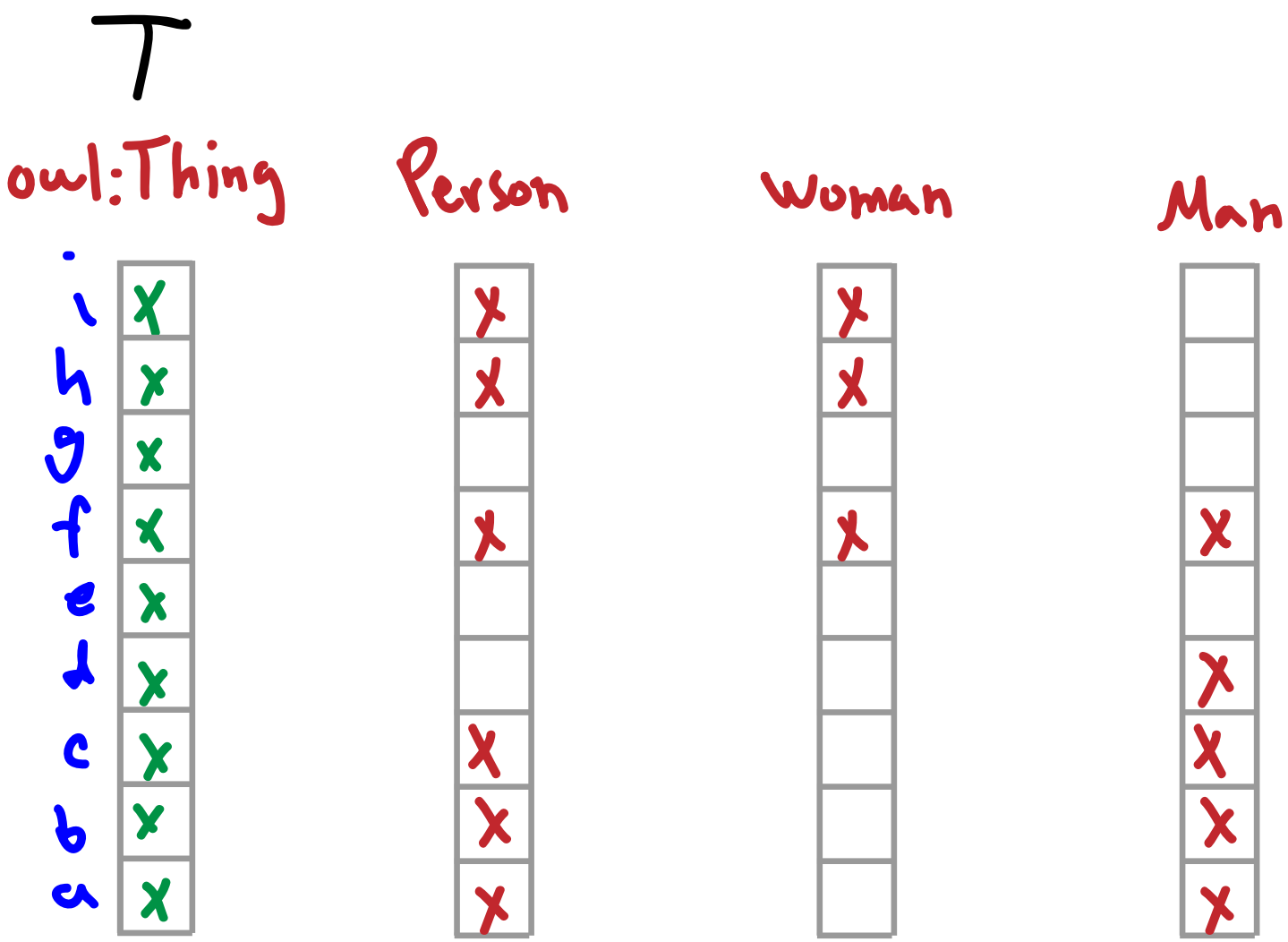

⊤
owl:Thing
i
h
g
f
e
d
c
b
a

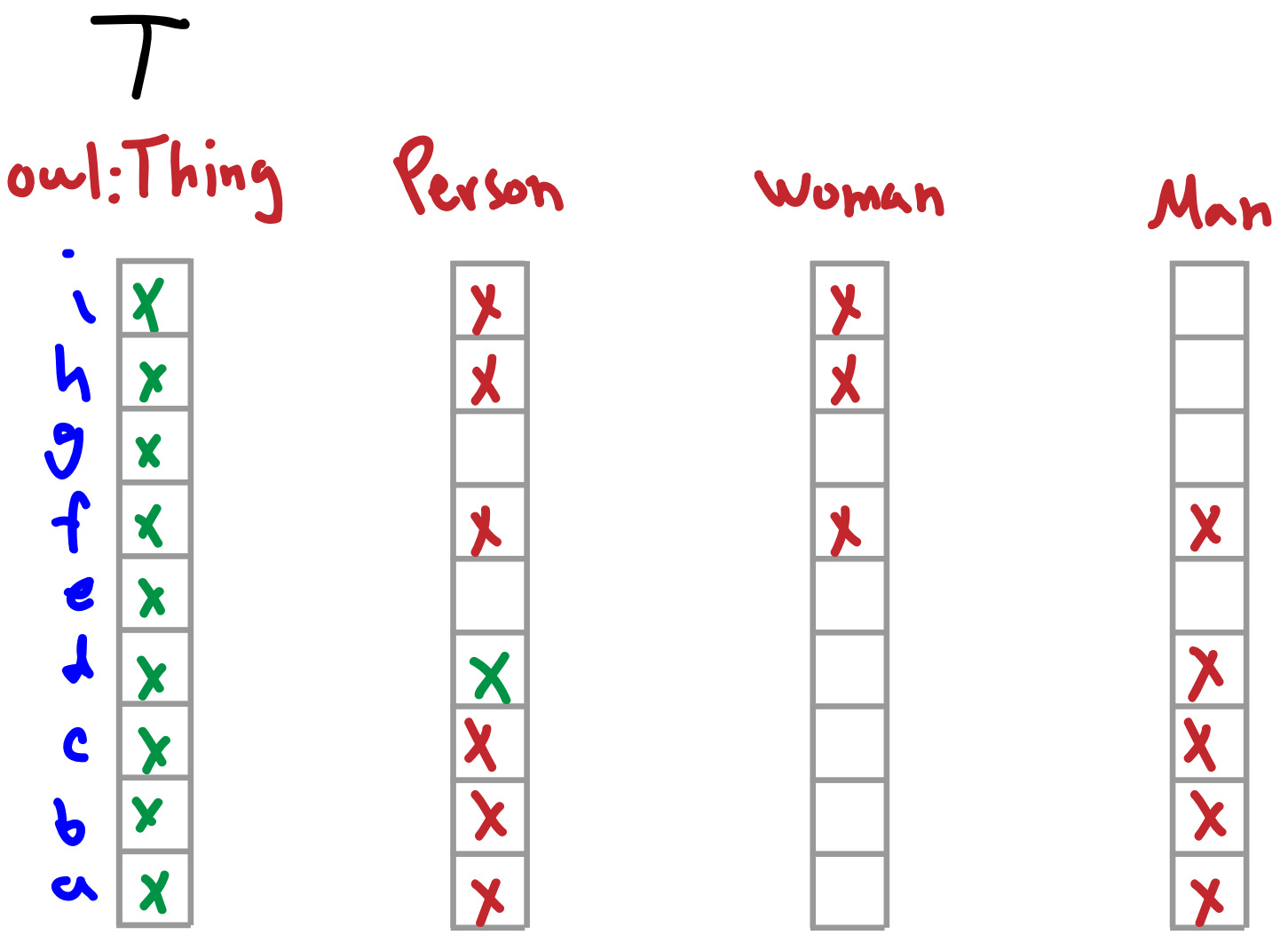

⊤
⊥
owl:Thing
owl:Nothing
i
h
g
f
e
d
c
b
a

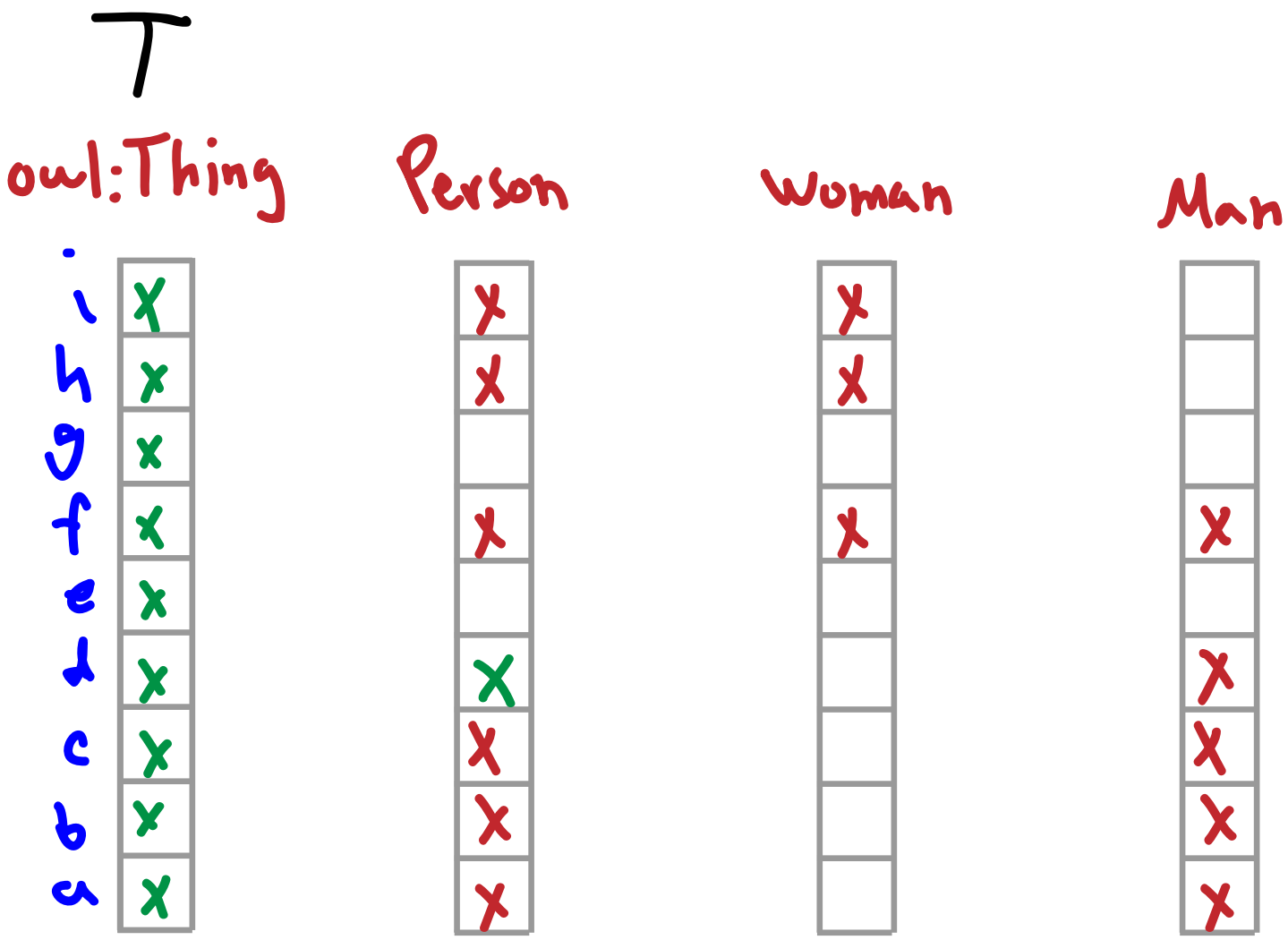

⊤
⊥
owl:Thing
owl:Nothing
Person
i
h
g
f
e
d
c
b
a


⊤
⊥
owl:Thing
owl:Nothing
Person
Woman
i
h
g
f
e
d
c
b
a


⊤
owl:Thing
Person
Woman
Man
i
h
g
f
e
d
c
b
a


⊤
owl:Thing
Person
Woman
Man
i
h
g
f
e
d
c
b
a
Man ⊑ Person
subclass


⊤
owl:Thing
Person
Woman
Man
i
h
g
f
e
d
c
b
a
Man ⊑ Person
subclass

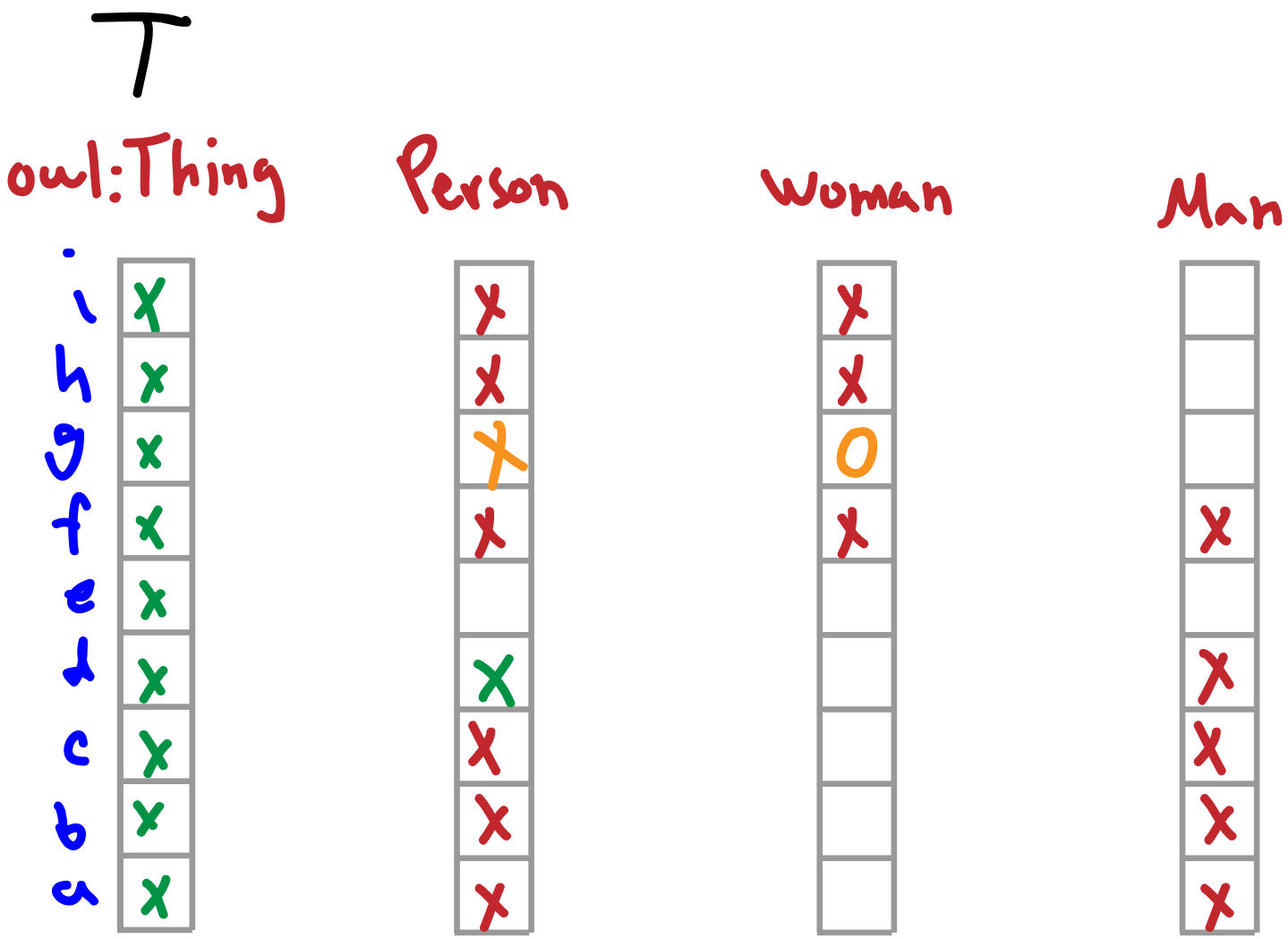


$$\text{Person} \equiv \text{Man} \sqcup \text{Woman}$$

equivalent union

$$\text{Person} \equiv \text{Man} \sqcup \text{Woman}$$

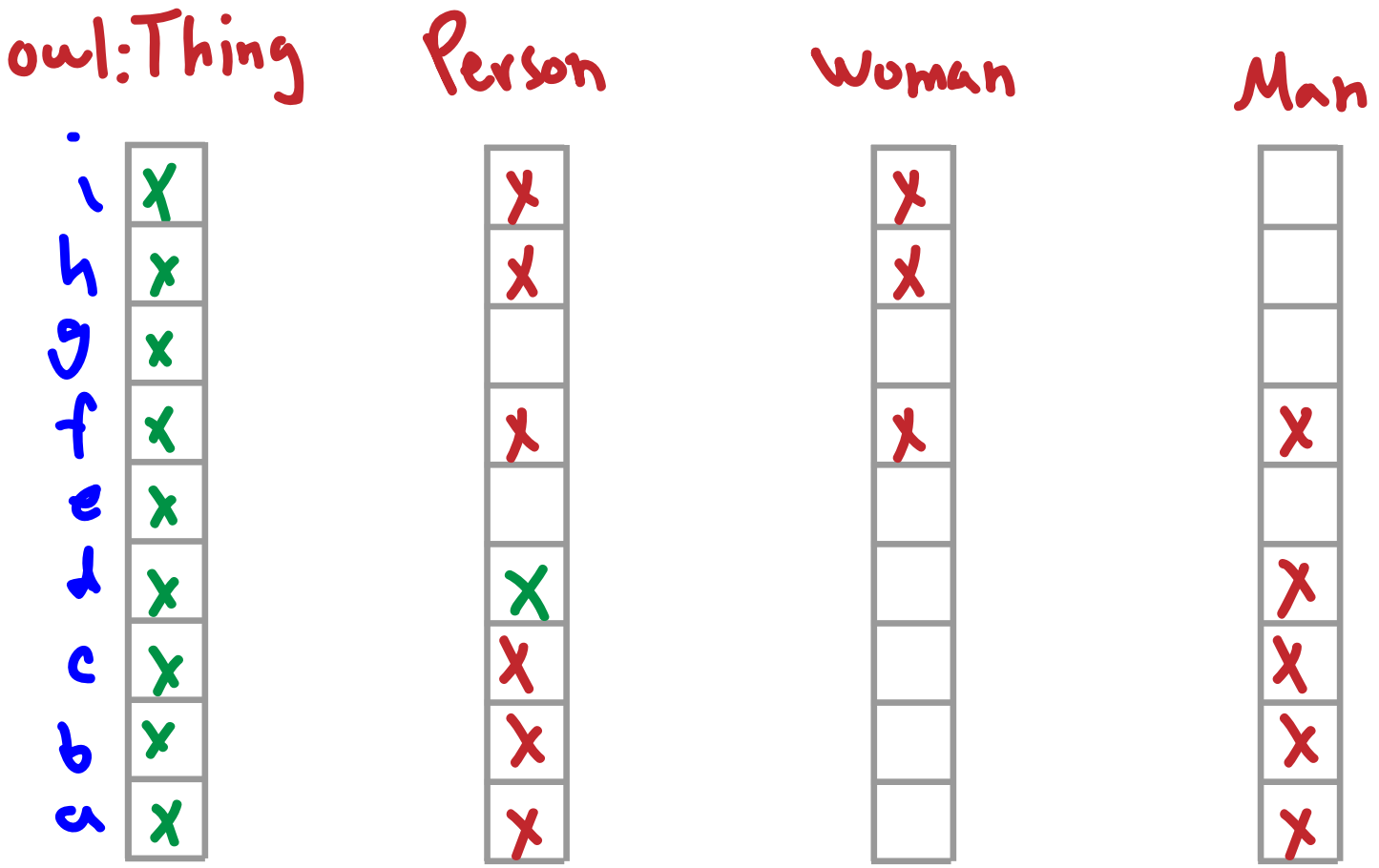
equivalent union



$$\text{Person} \equiv \text{Man} \sqcup \text{Woman}$$

$$\{g\} \sqcap \text{Woman} \sqsubseteq \bot$$

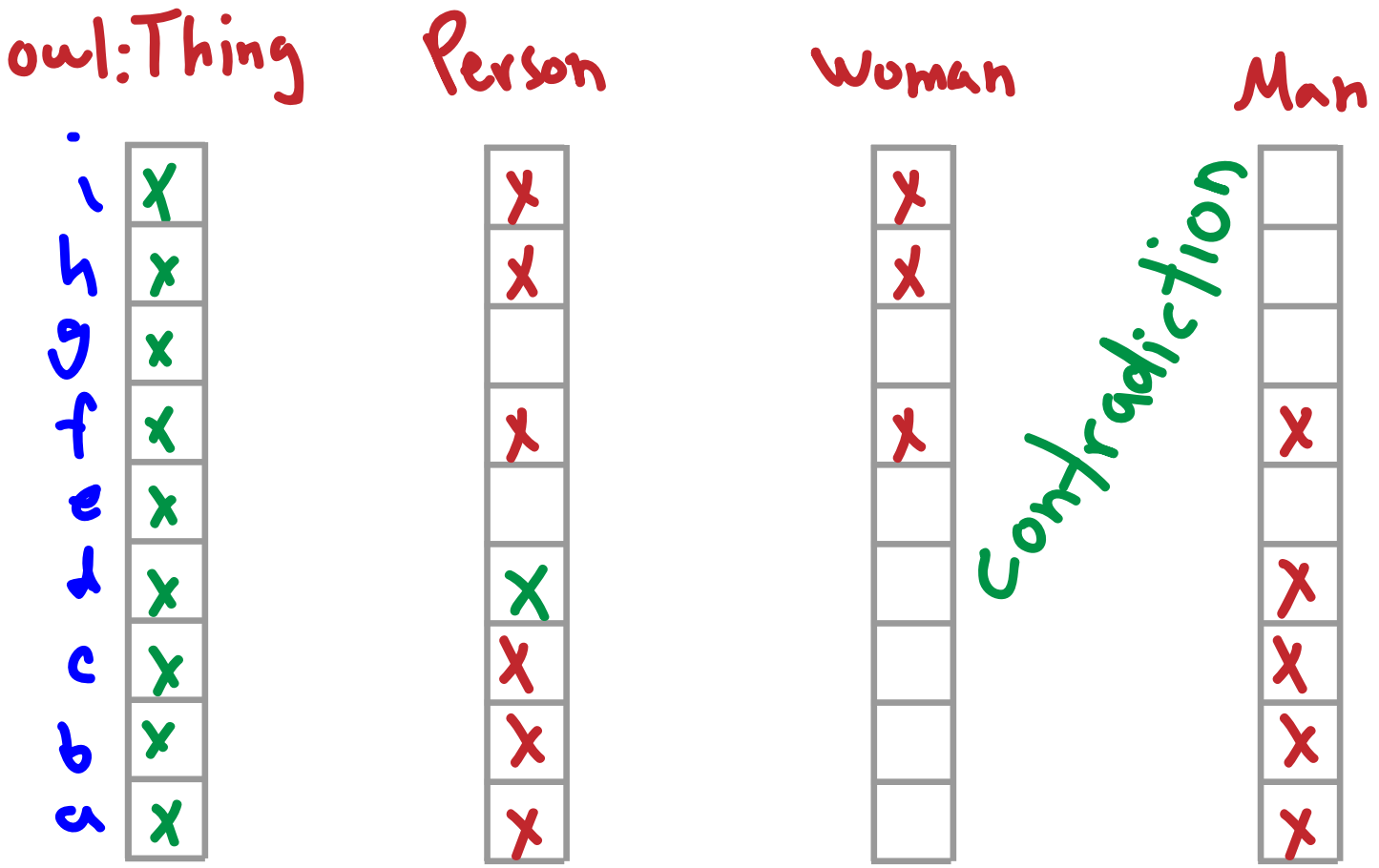
$\top$



$\mathrm{Man} \sqsubseteq \mathrm{Person}$

$\mathrm{Man} \sqcap \mathrm{Woman} \sqsubseteq \bot$

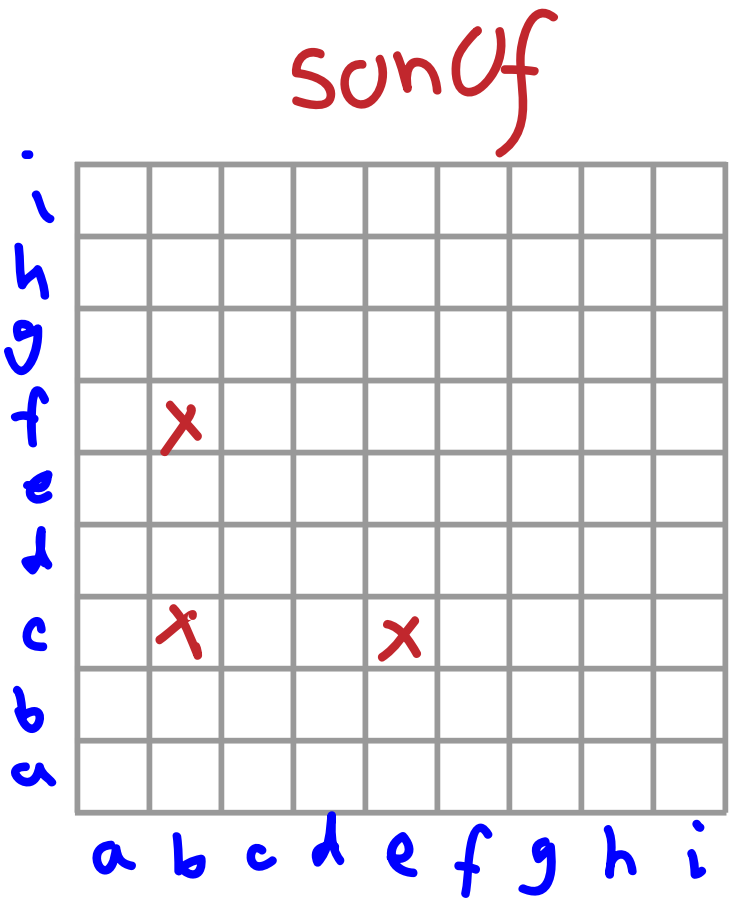
$\top$



$$Man \sqsubseteq Person$$

$$Man \sqcap Woman \sqsubseteq \bot$$

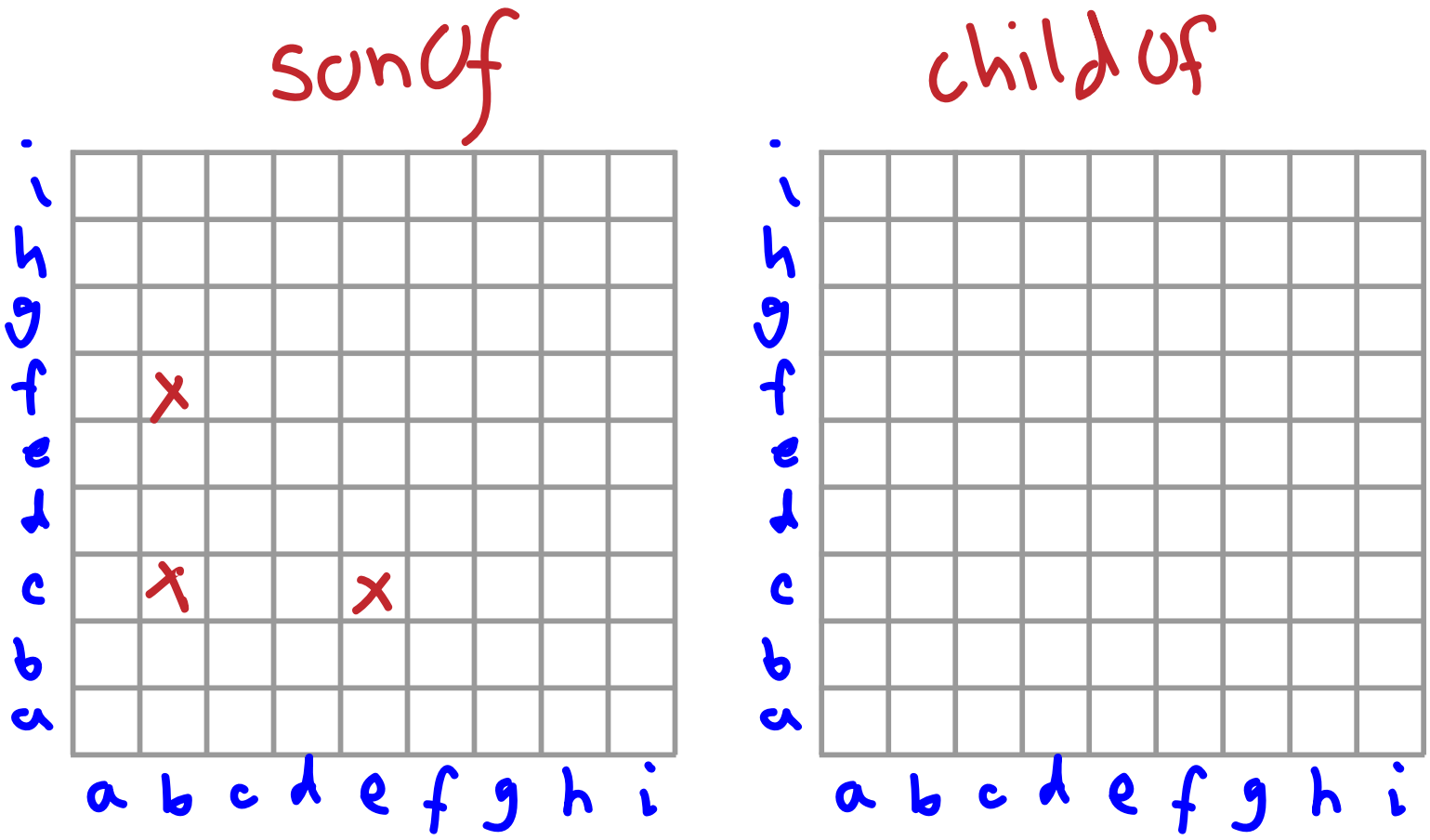
Properties



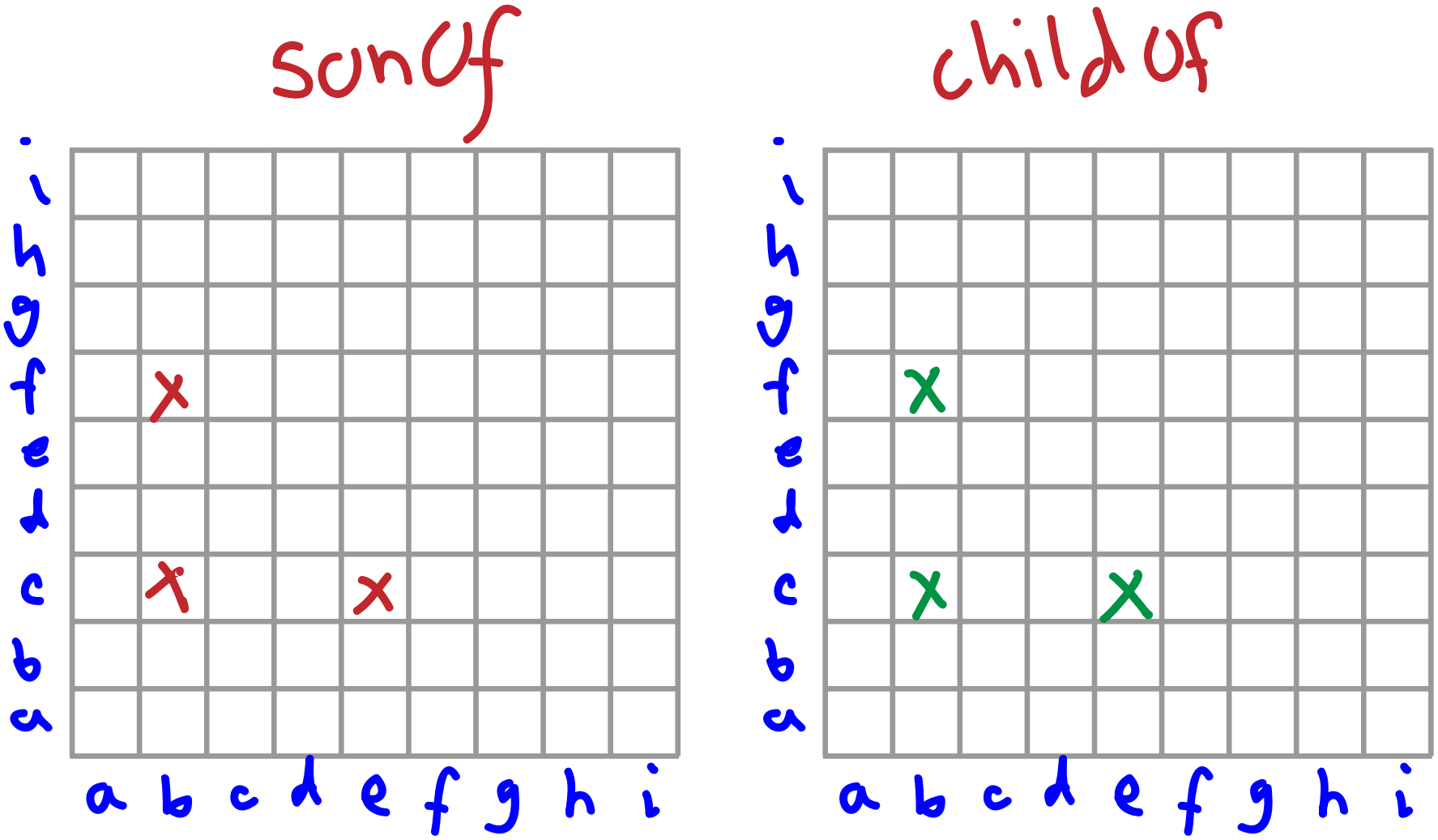
b son of c

b son of f

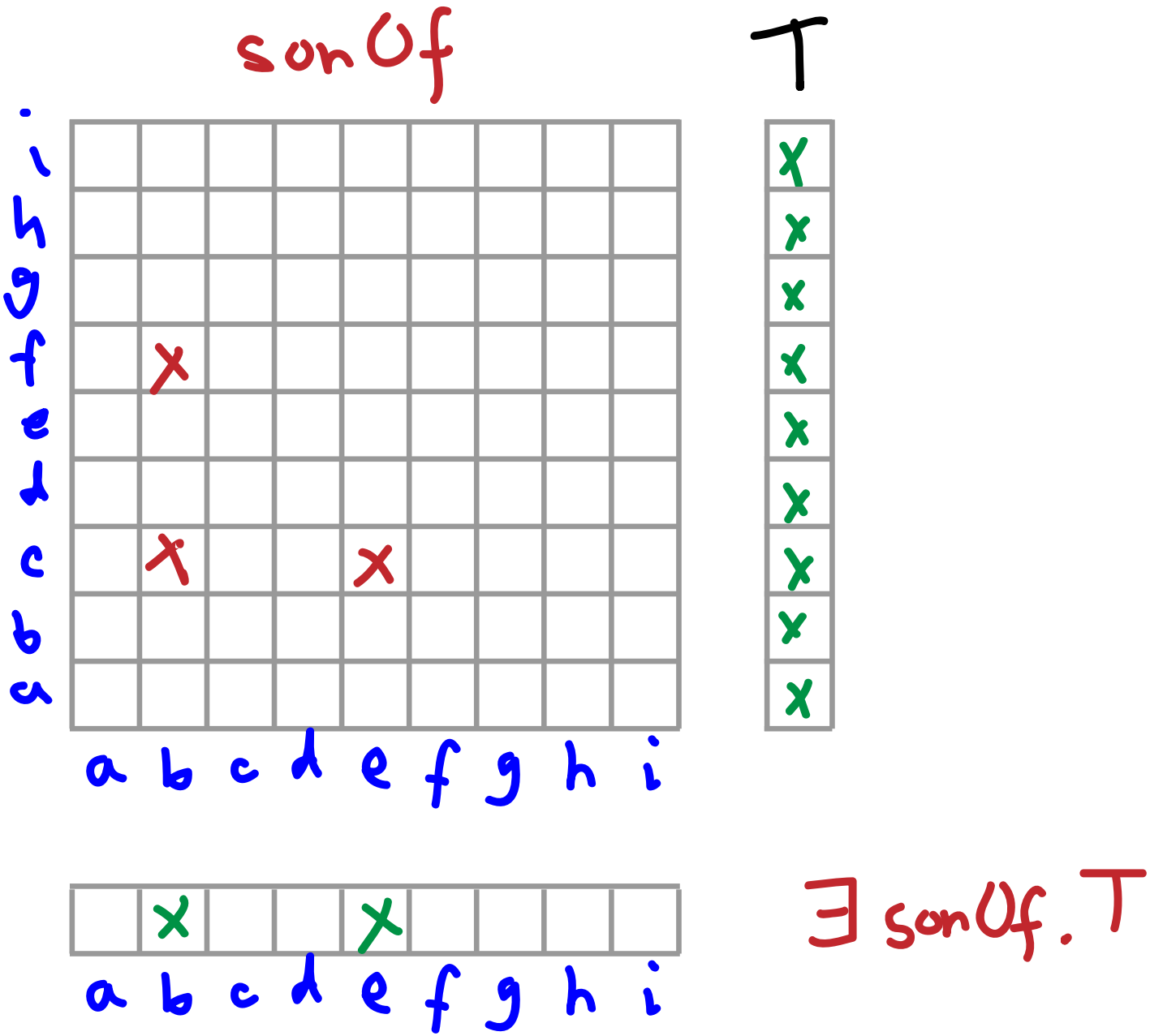
e son of c

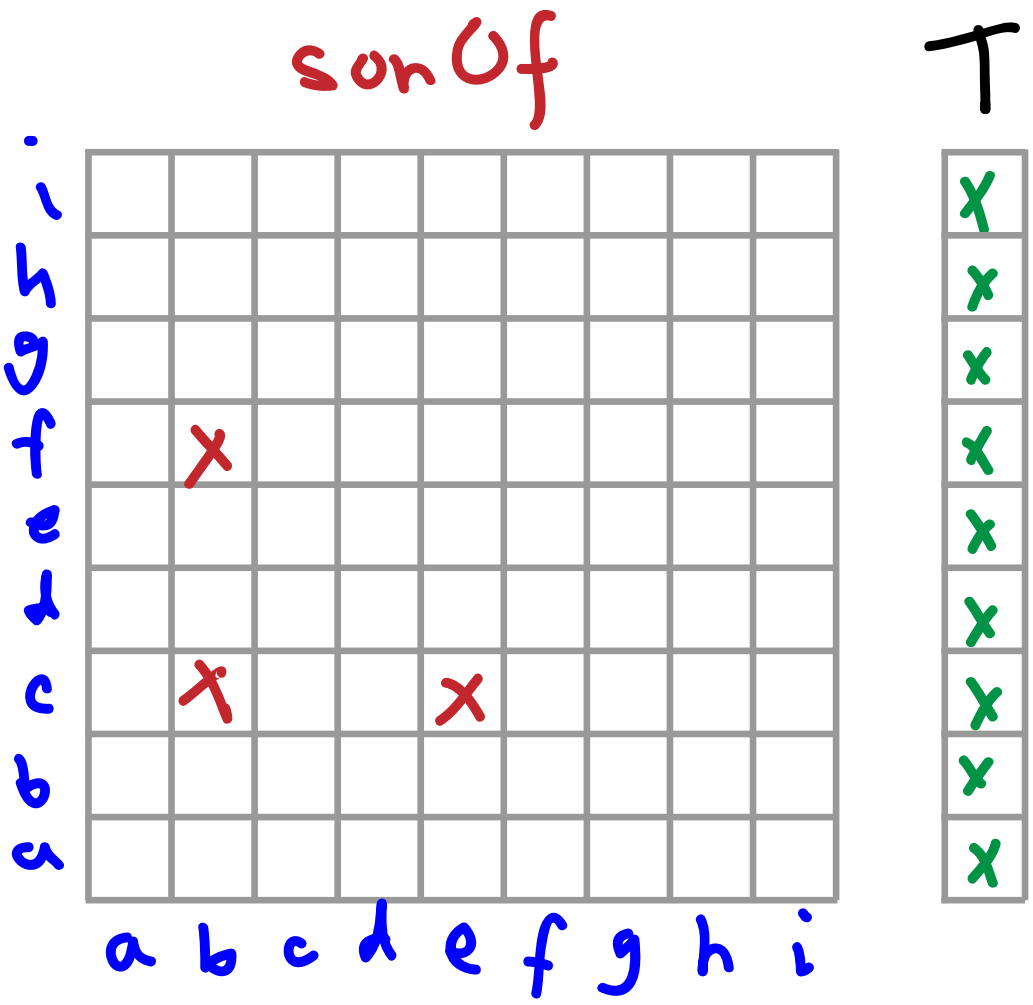


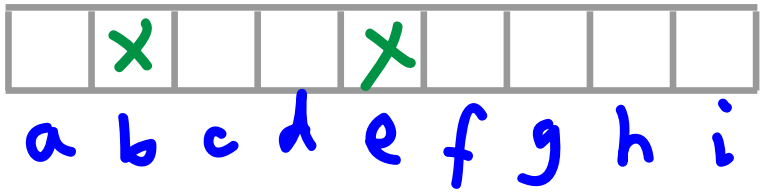
sonOf ⊑ childOf



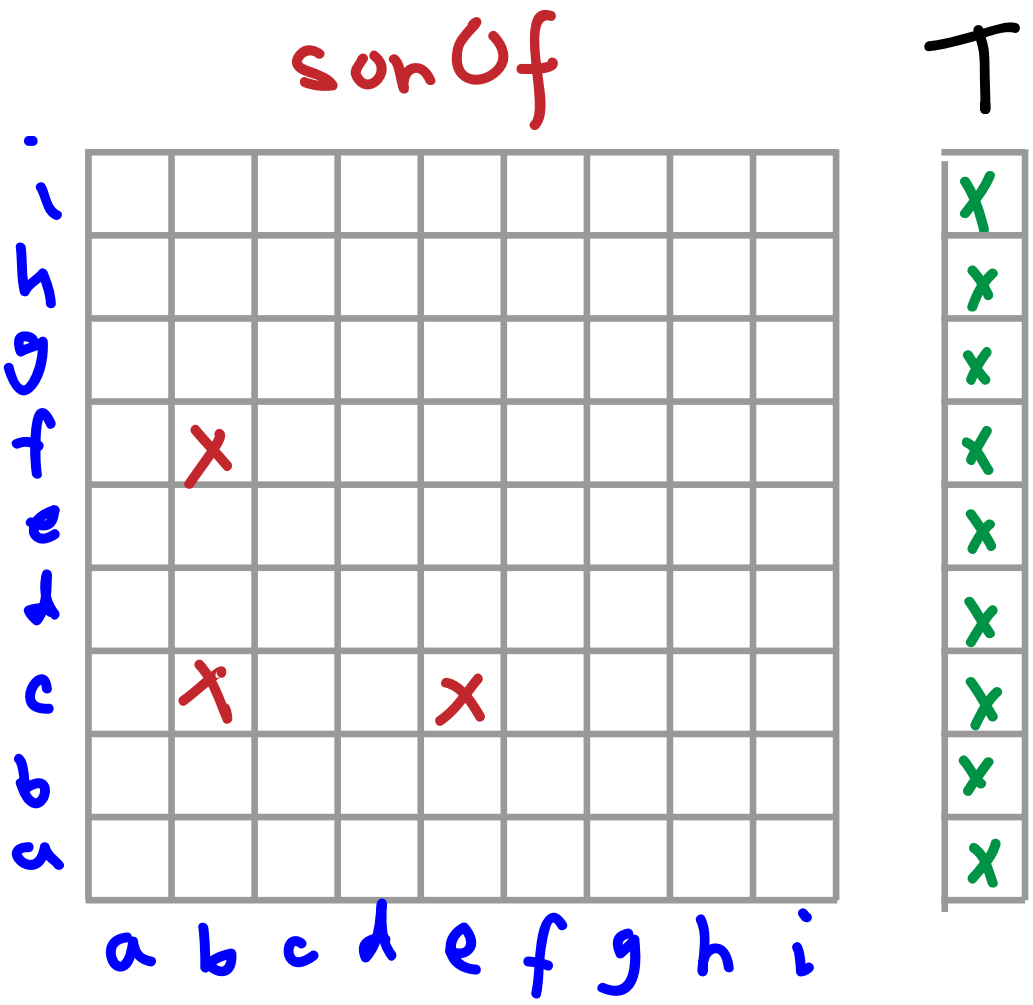
sonOf ⊑ childOf

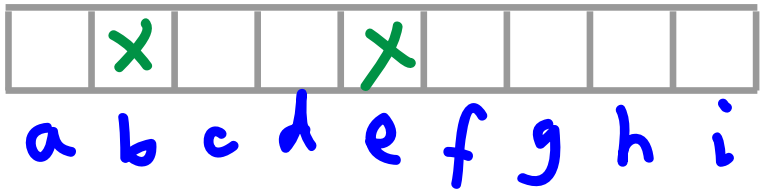

sonOf
T
i
h
g
f
e
d
c
b
a
a b c d e f g h i
a b c d e f g h i
∃ sonOf. T





$\exists \, sonOf.\top \sqsubseteq Person$



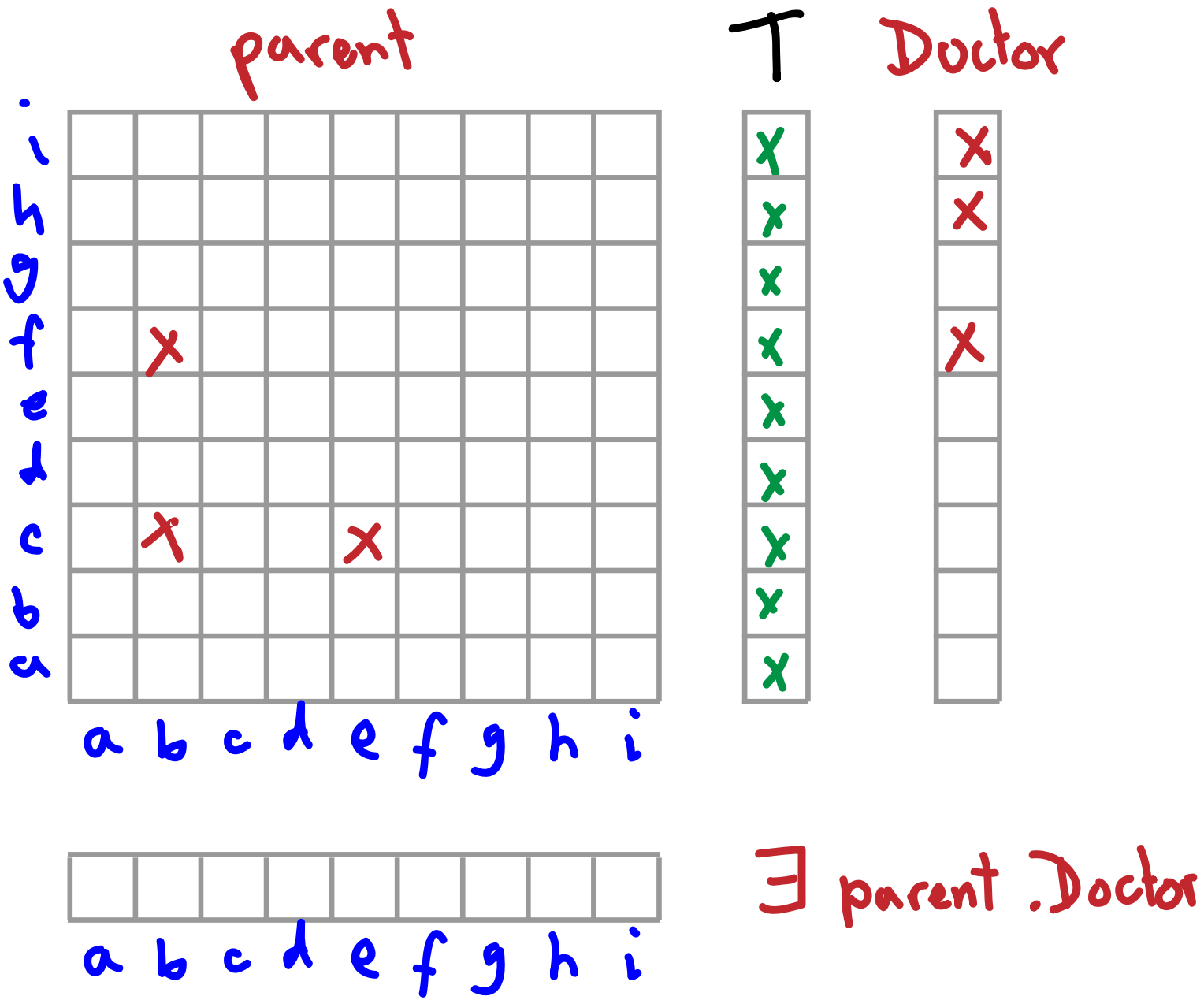


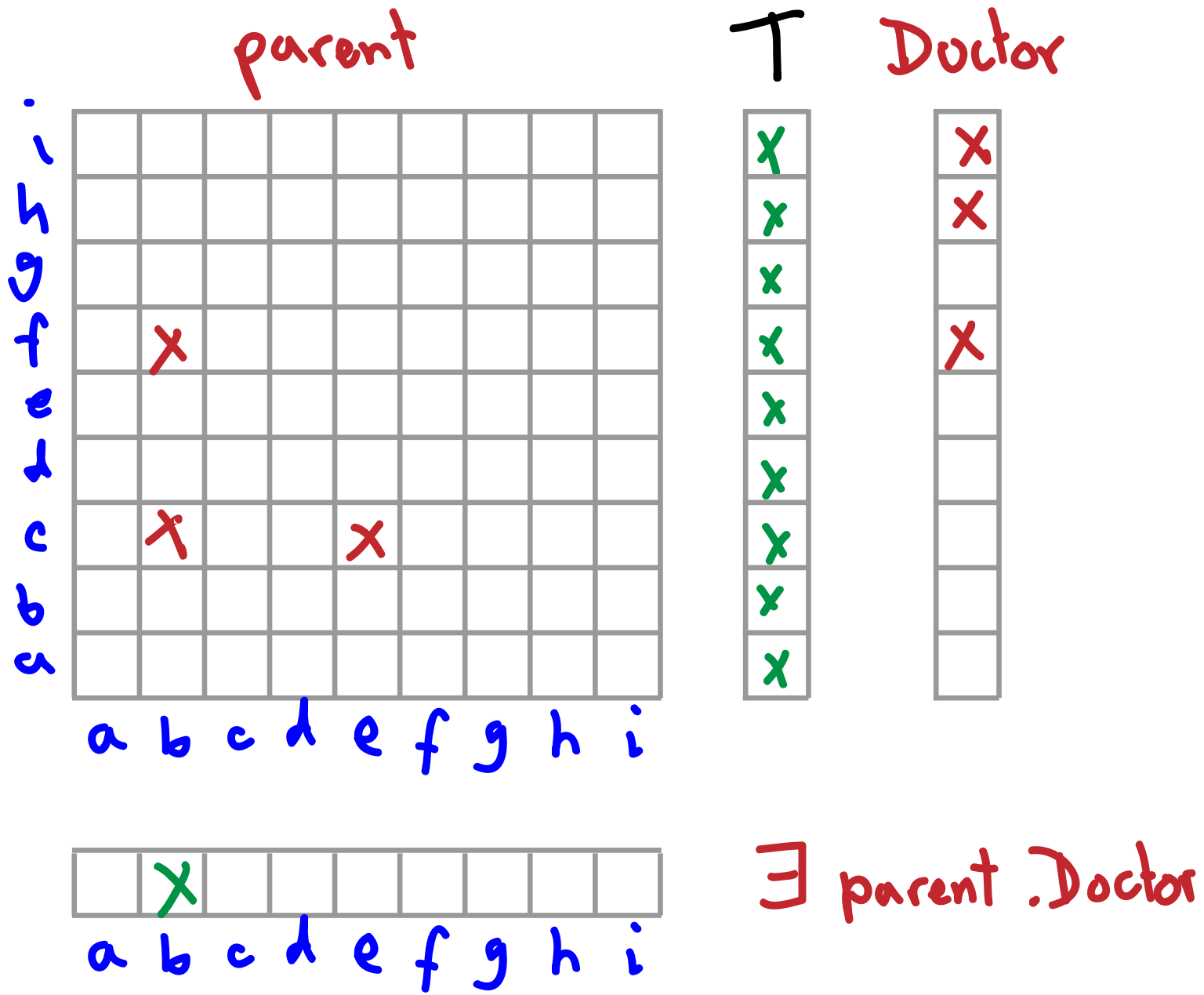
$\exists\, sonOf.\top \sqsubseteq Person$

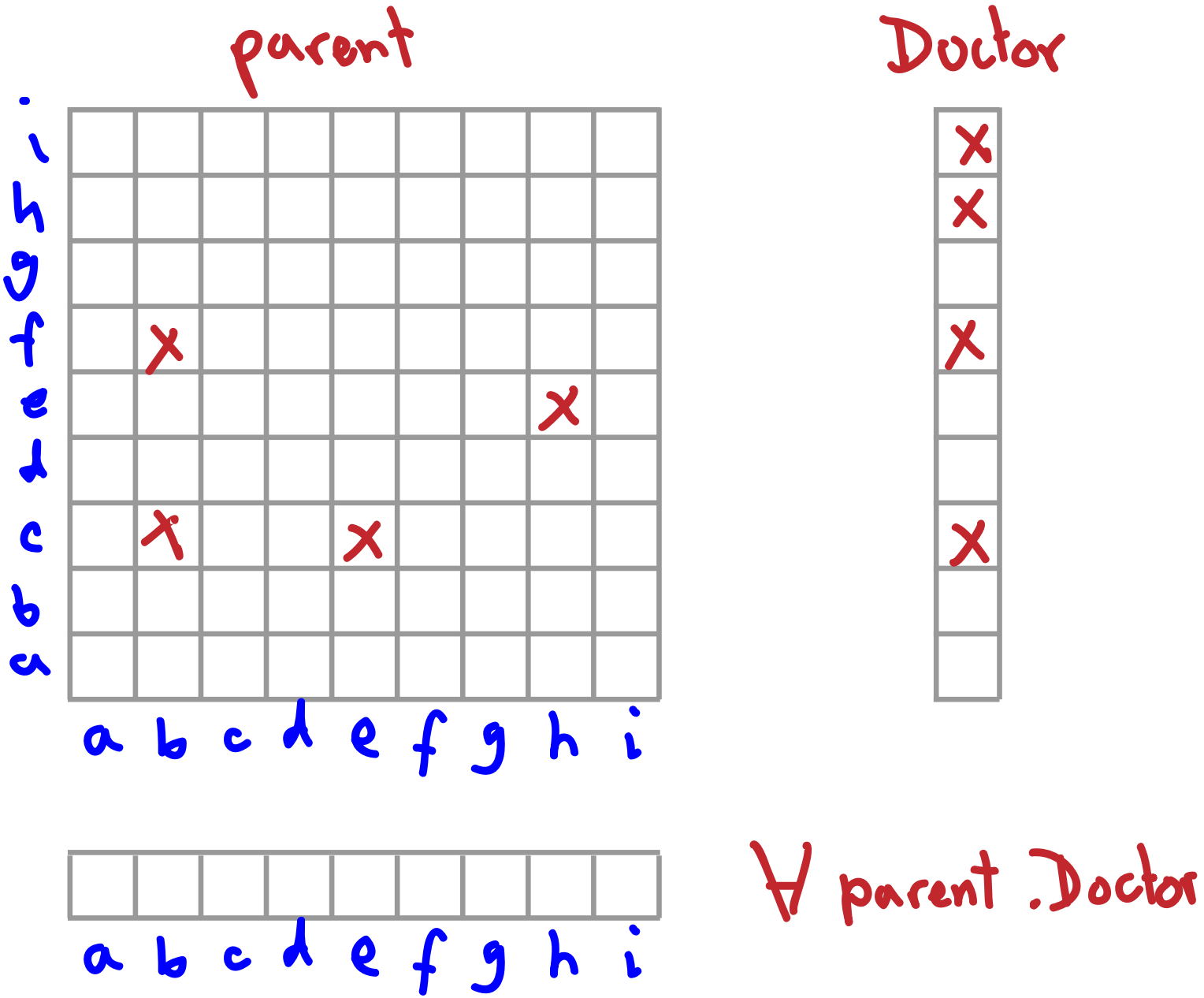
$domain(sonOf) = Person$

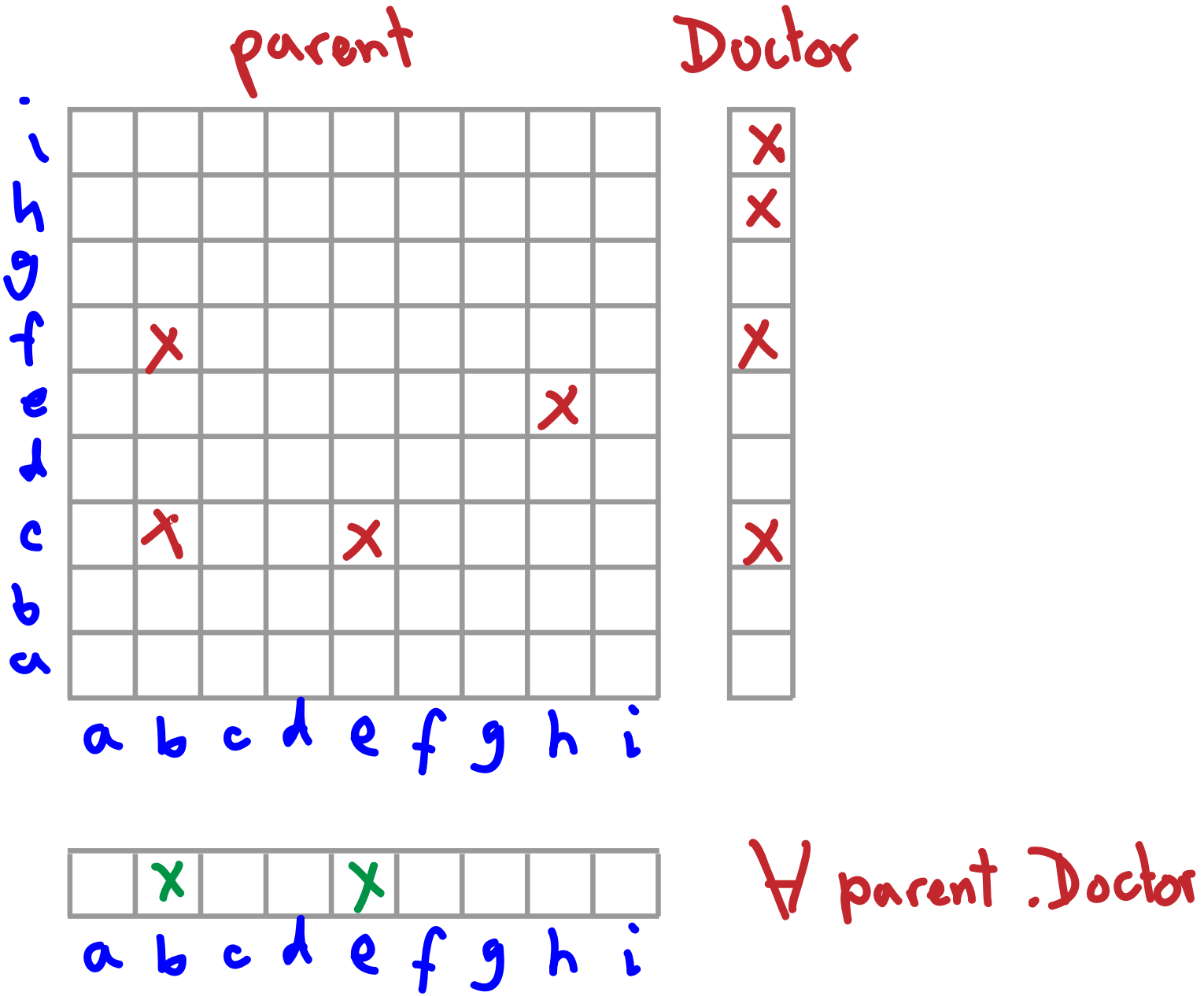
P rdfs:domain C

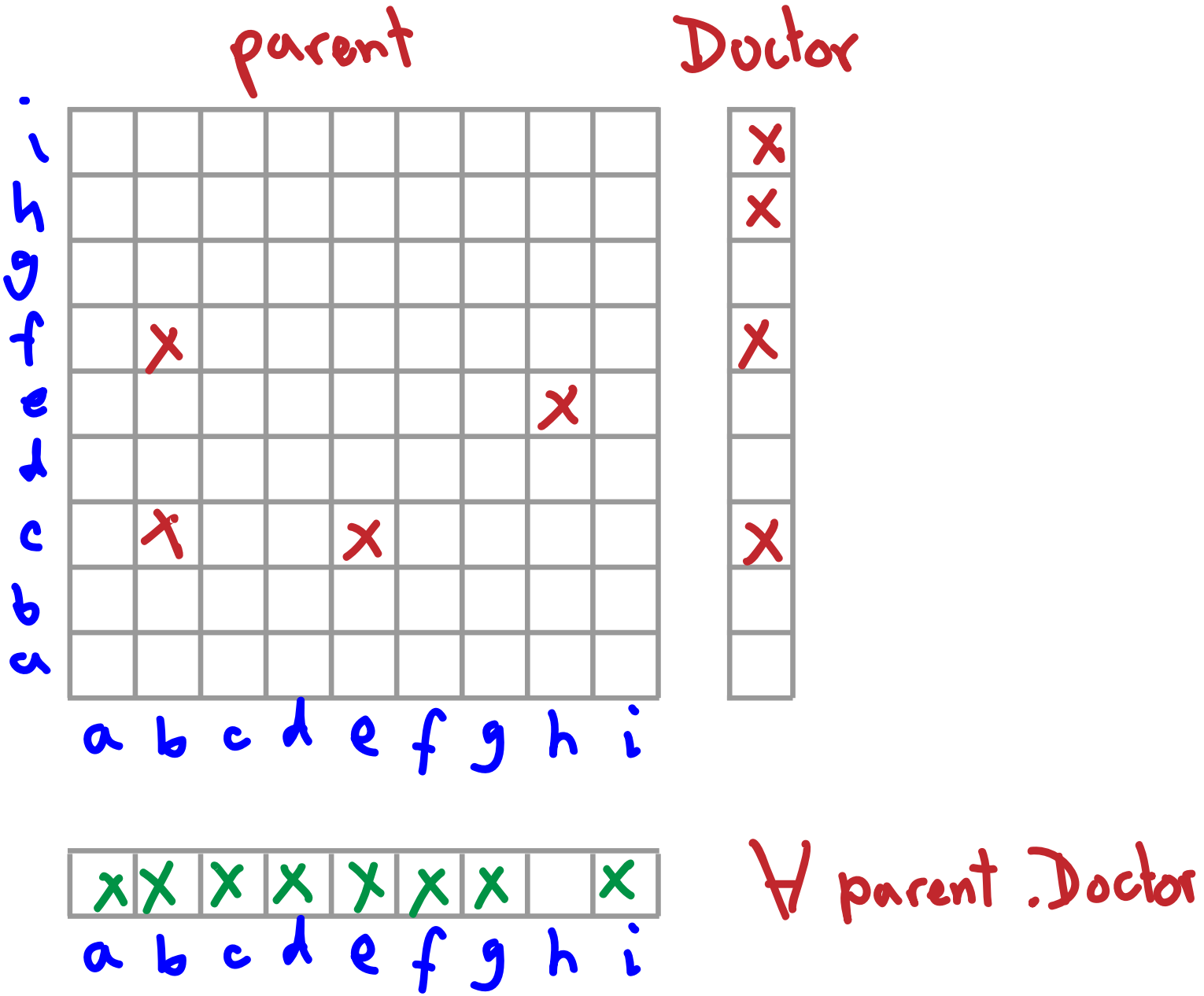
$\exists P.\top \sqsubseteq C$

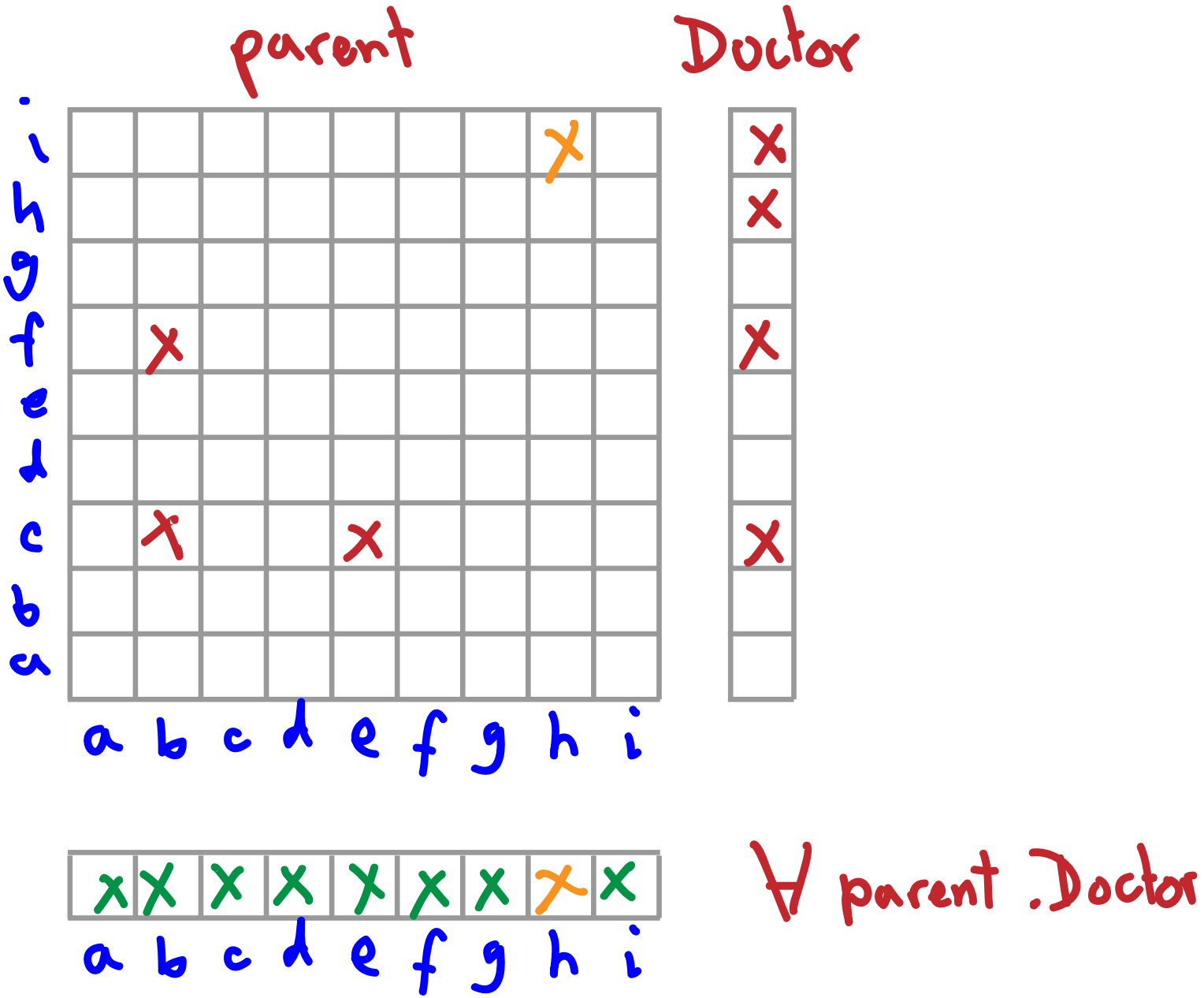

parent
⊤
Doctor
i
h
g
f
e
d
c
b
a
a b c d e f g h i
a b c d e f g h i
∃ parent . Doctor

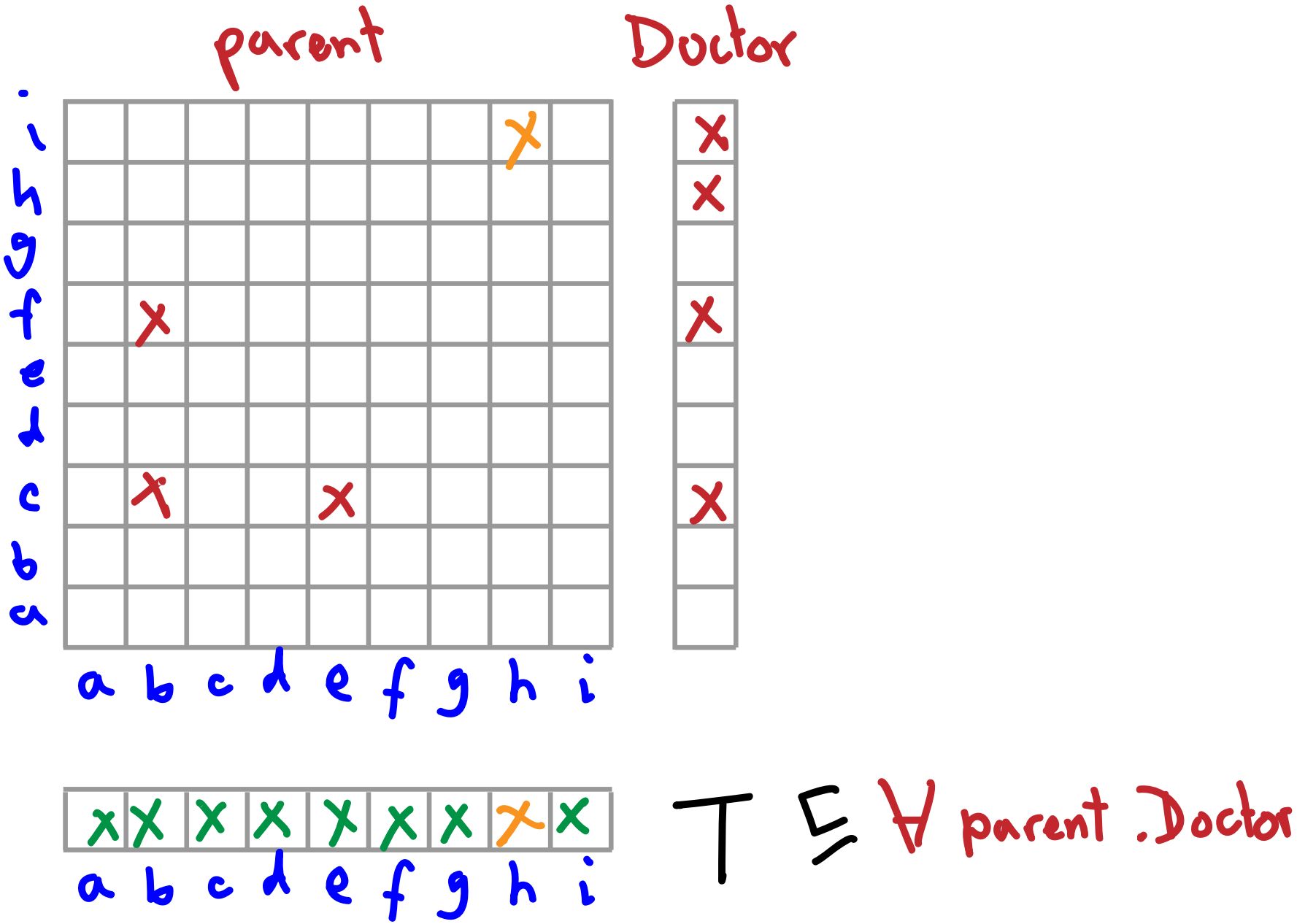

parent
⊤
Doctor
i
h
g
f
e
d
c
b
a
a b c d e f g h i
∃ parent . Doctor

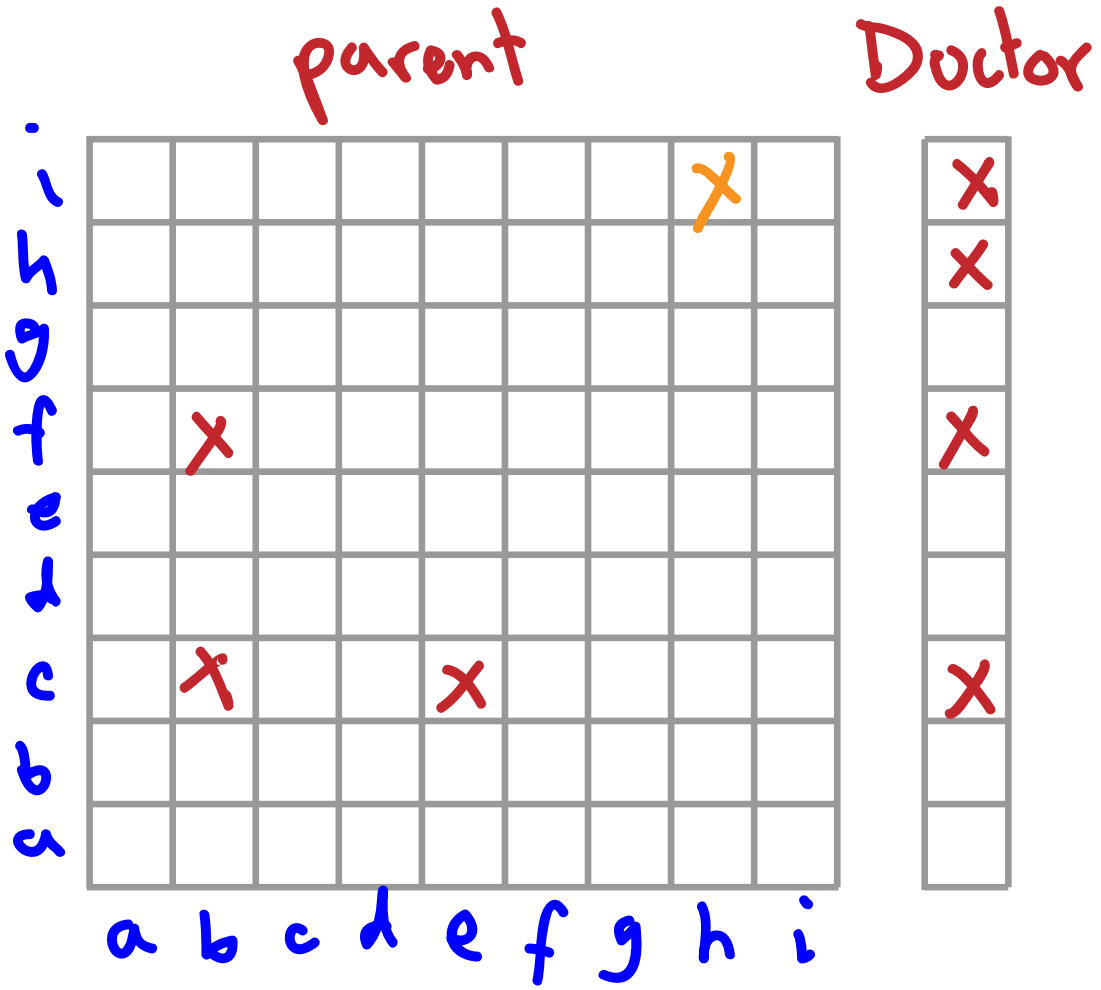

parent
Doctor
i
h
g
f
e
d
c
b
a
a b c d e f g h i
a b c d e f g h i
∀ parent . Doctor

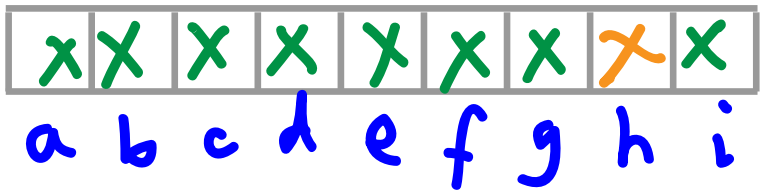

parent
Doctor
i
h
g
f
e
d
c
b
a
a b c d e f g h i
a b c d e f g h i
∀ parent . Doctor


parent
Doctor
i
h
g
f
e
d
c
b
a
a b c d e f g h i
a b c d e f g h i
∀ parent . Doctor


parent
Doctor
i
h
g
f
e
d
c
b
a
a b c d e f g h i
a b c d e f g h i
∀ parent . Doctor


parent
Doctor
i
h
g
f
e
d
c
b
a
a b c d e f g h i
a b c d e f g h i
⊤ ⊑ ∀ parent . Doctor





$\top \sqsubseteq \forall \text{parent}.\text{Doctor}$

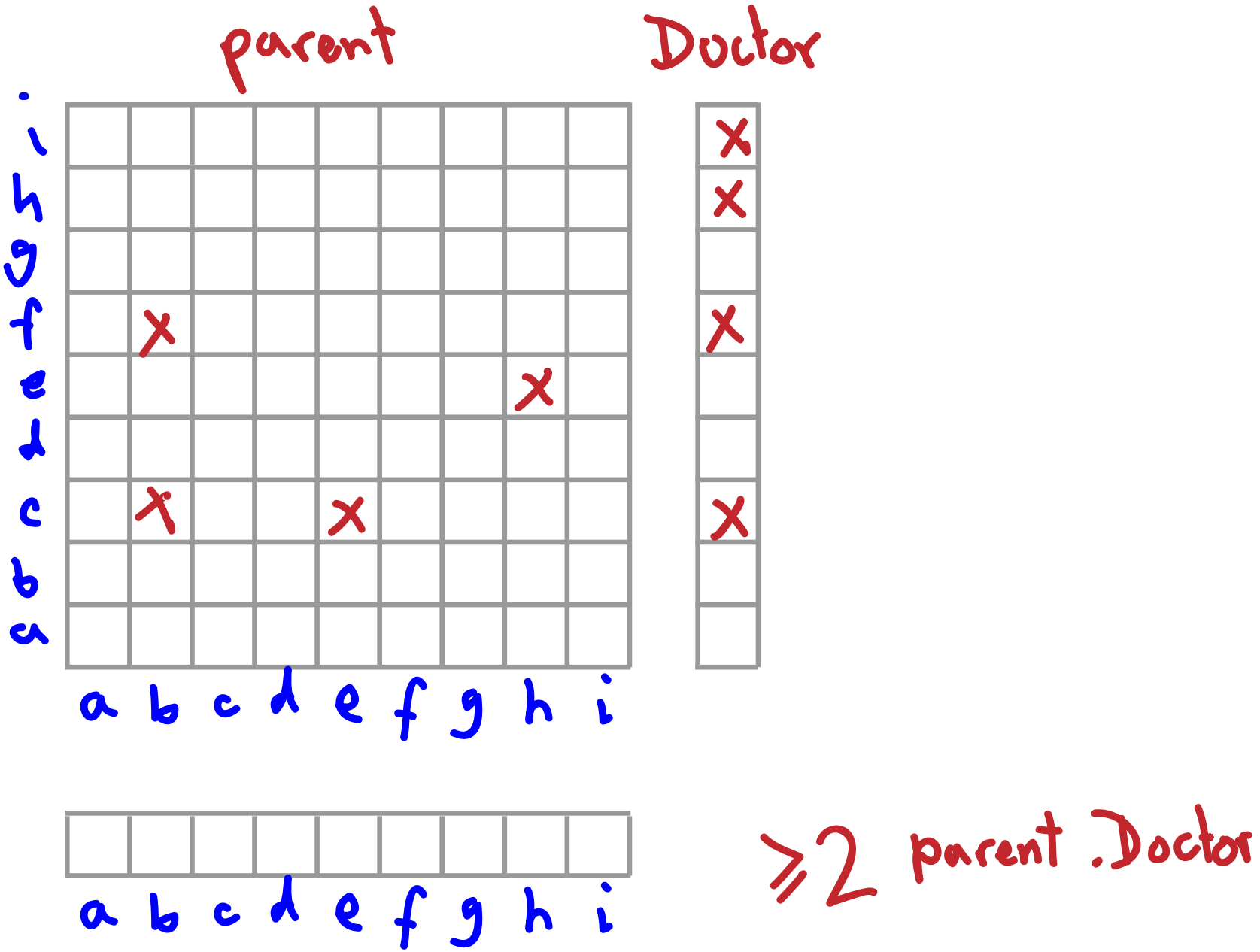
$\text{range}(\text{parent}) = \text{Doctor}$

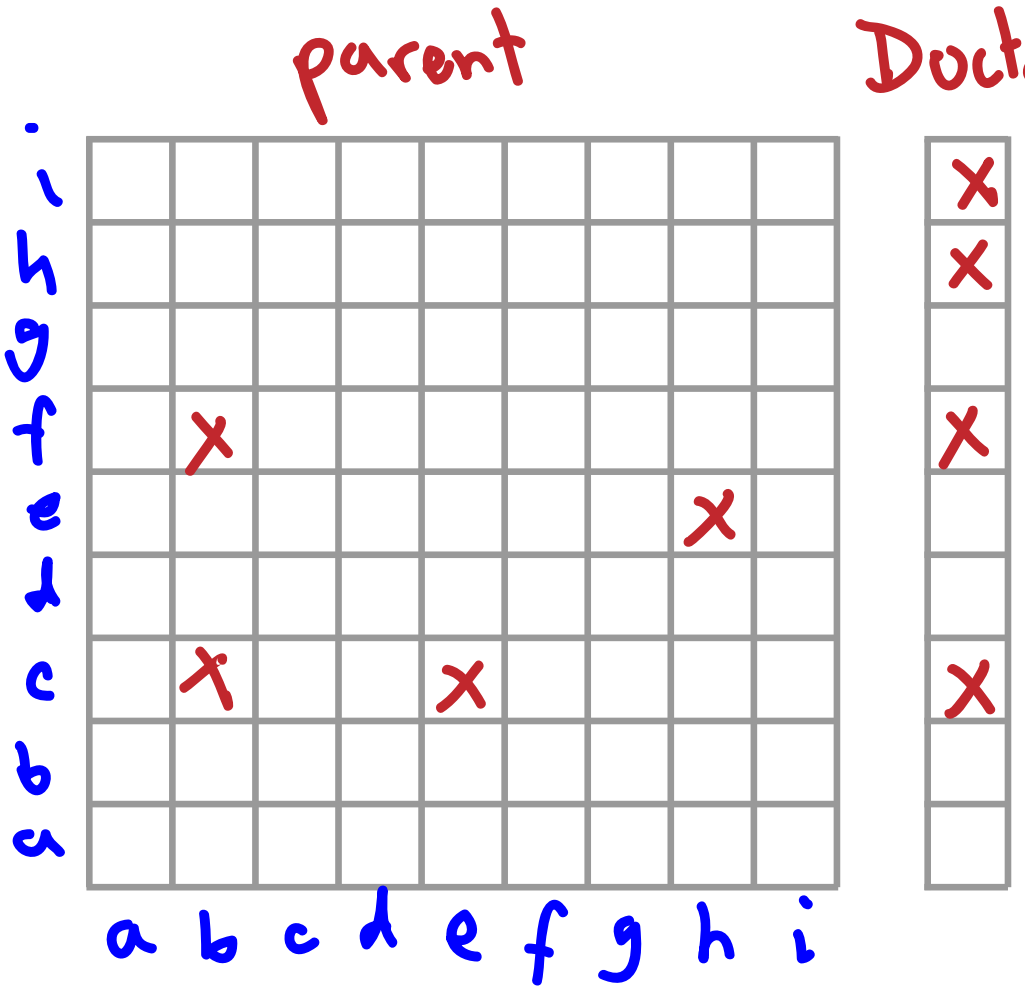
P rdfs:domain C

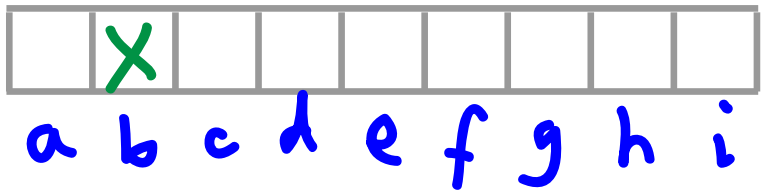
$\exists P.\top \sqsubseteq C$

P rdfs:range C

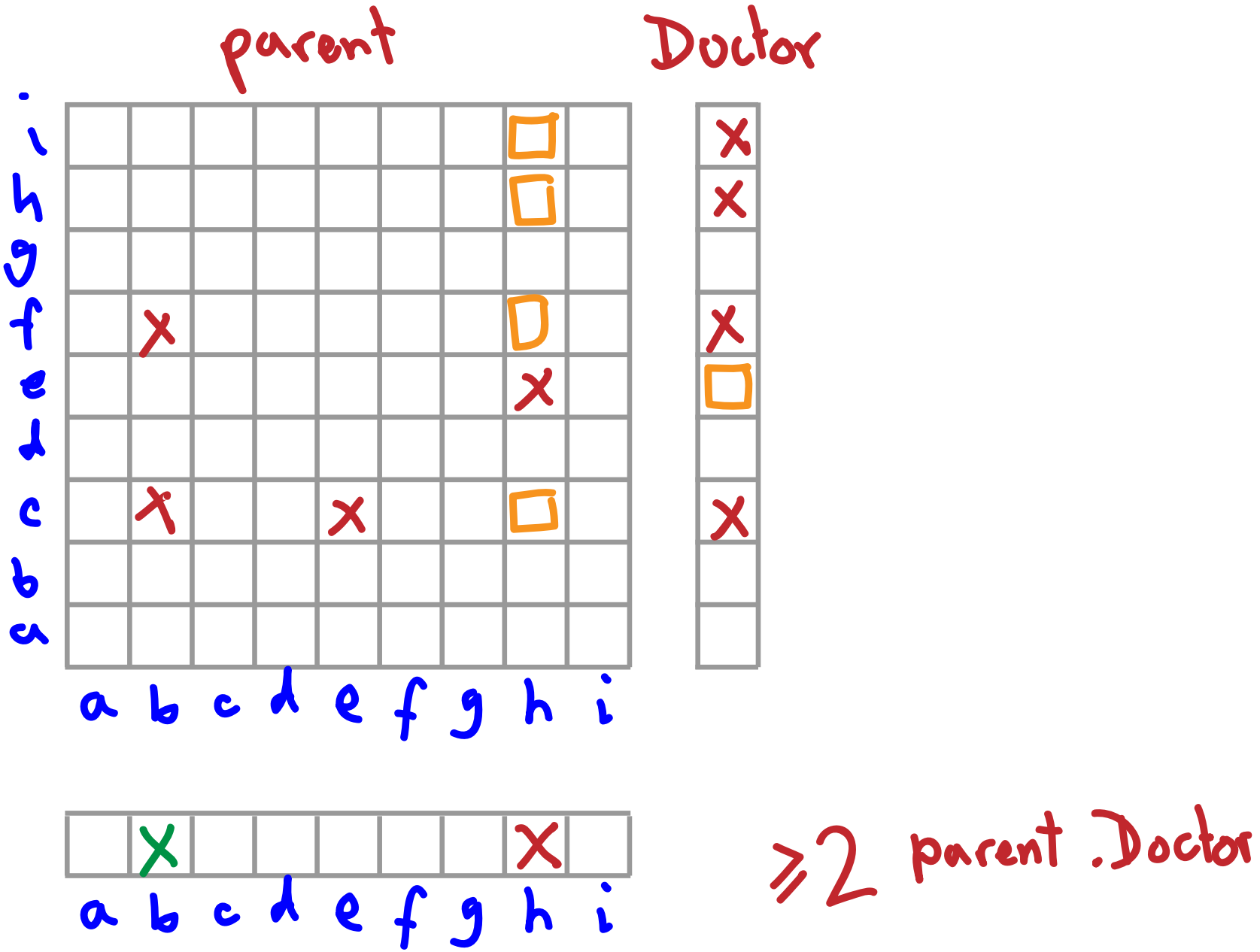
$\top \sqsubseteq \forall P.C$


parent
Doctor
i
h
g
f
e
d
c
b
a
a b c d e f g h i
a b c d e f g h i
≥2 parent .Doctor

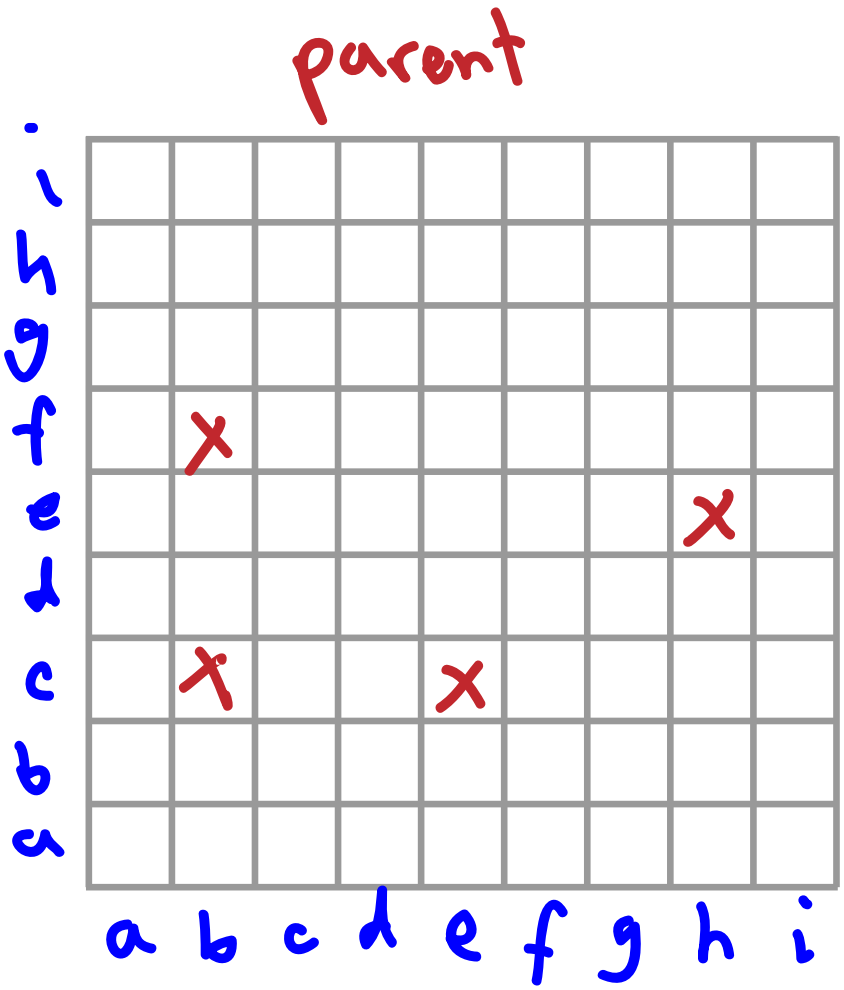


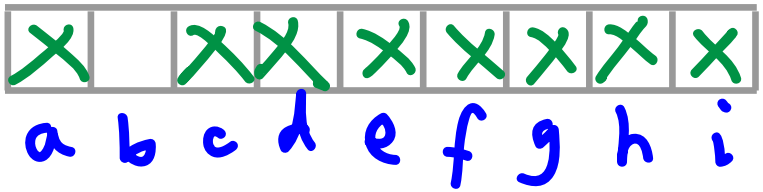


≥ 2 parent .Doctor

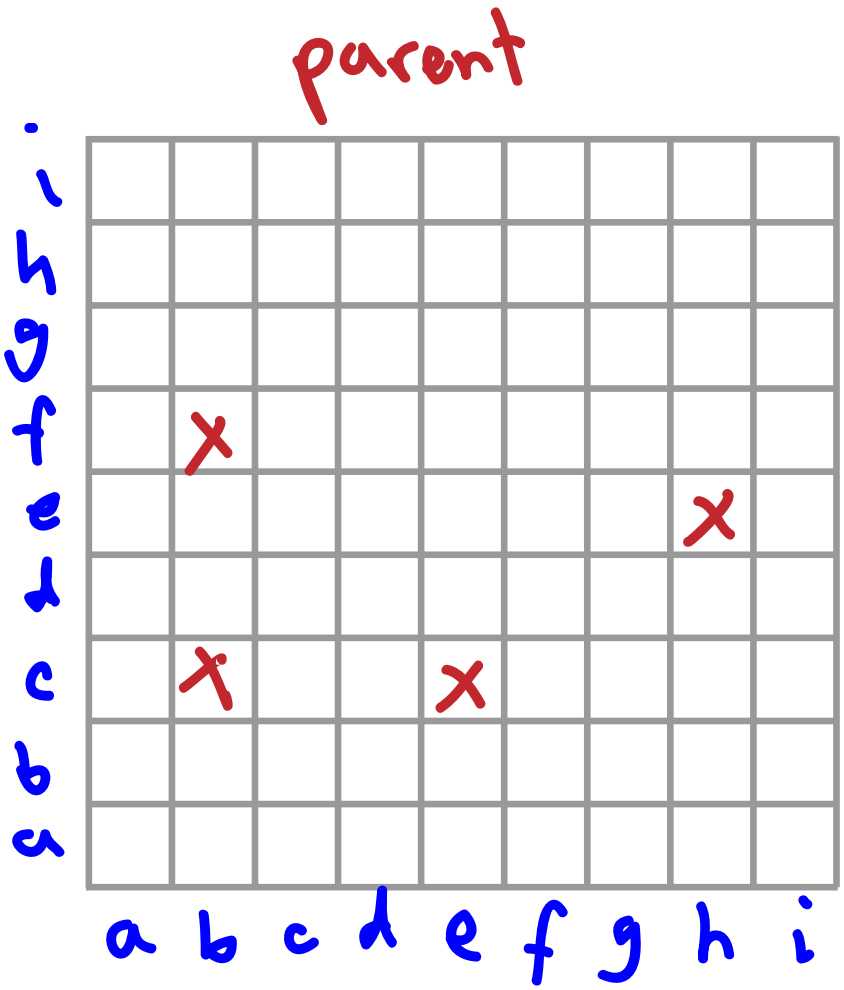


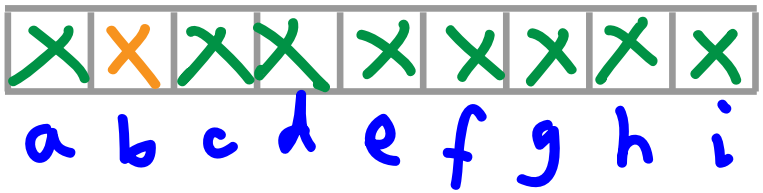
≥ 2 parent .Doctor



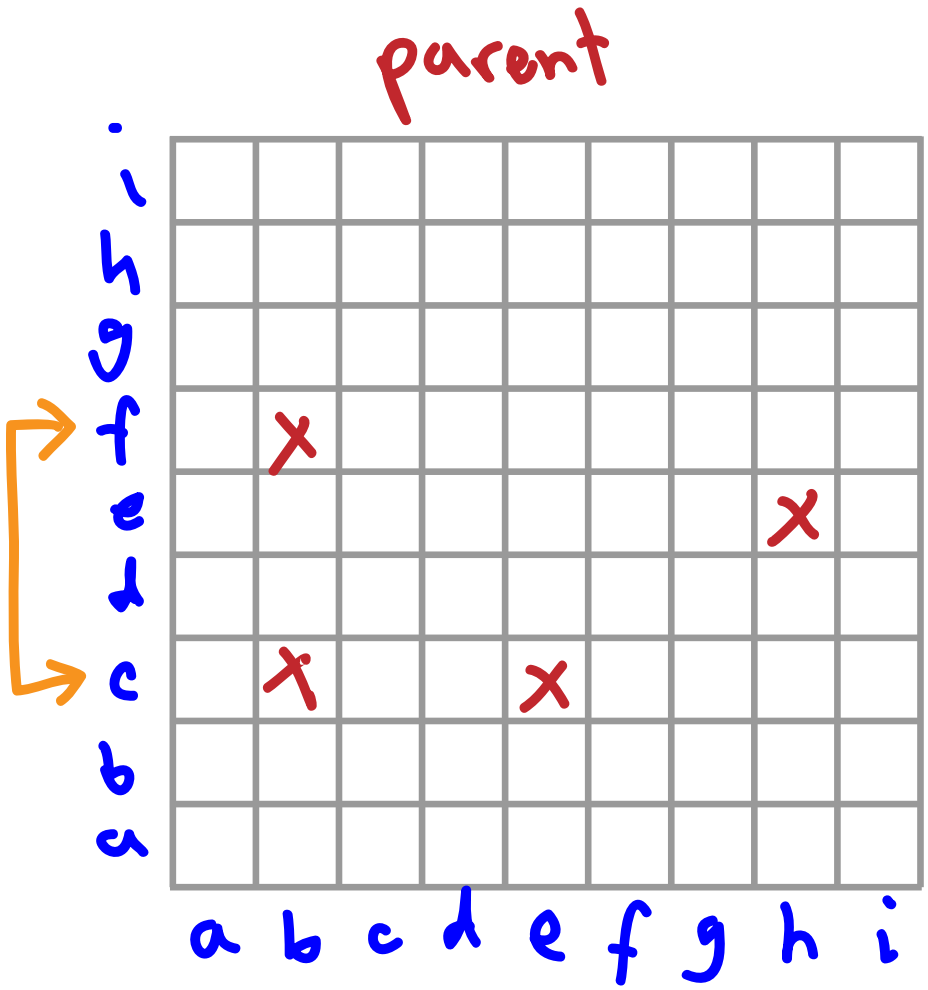


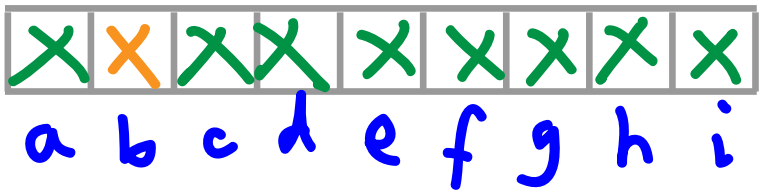
≤ 1 parent ✓





$T \sqsubseteq \leq 1\ \text{parent}.T$

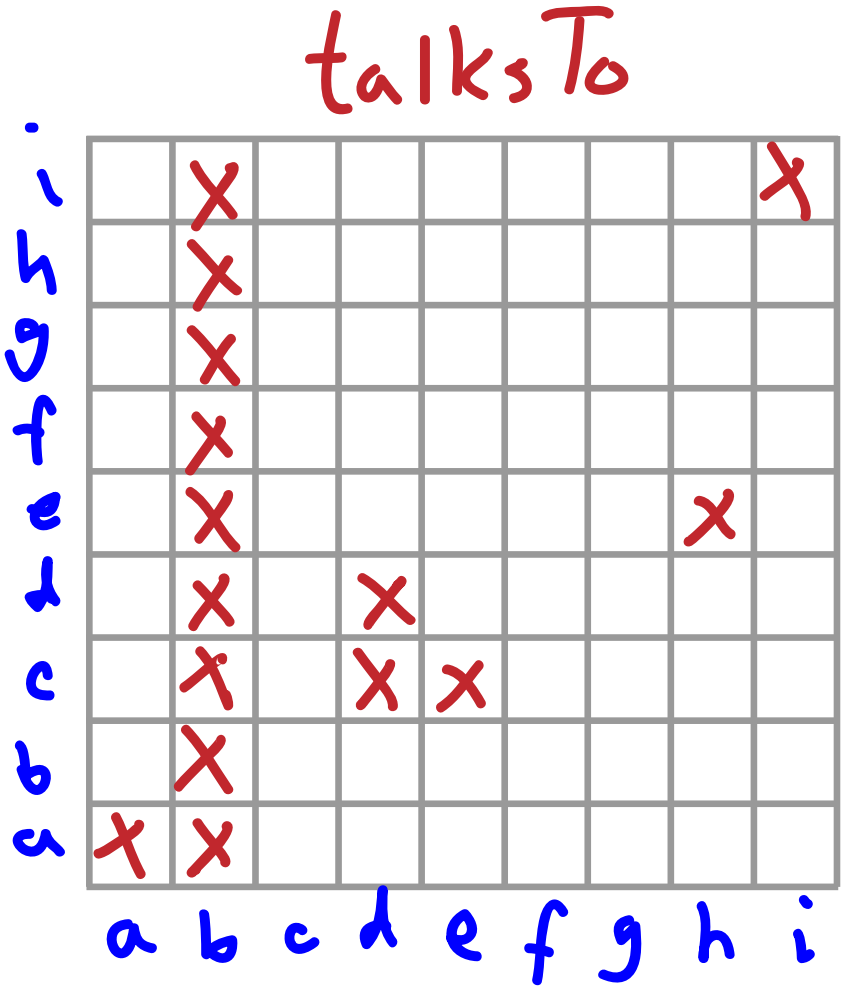


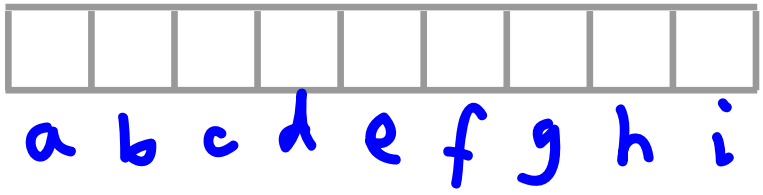


$f \simeq c$

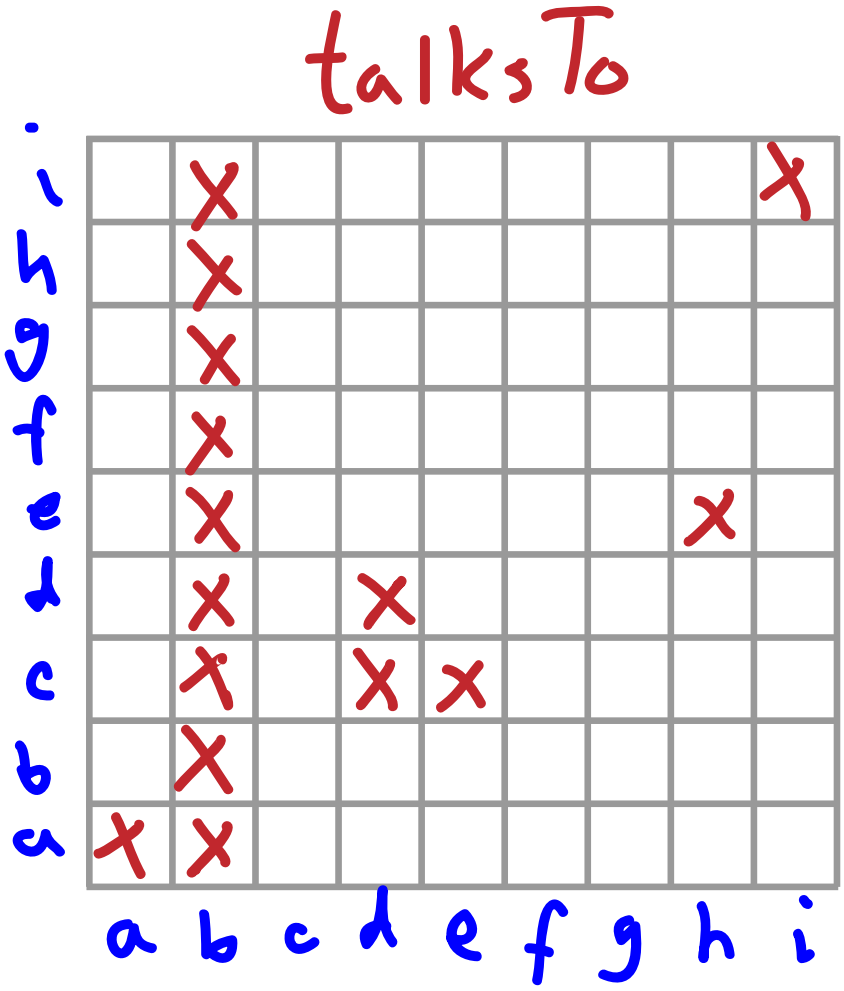
f owl:sameAs c

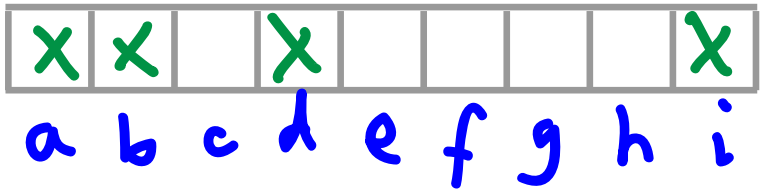
$\top \sqsubseteq \leq 1$ parent $.\top$





$\exists$ talksTo. Self





$\exists$ talksTo. Self

CLASSES

$\top$

$\bot$

$C_1 \sqcap C_2$

$C_1 \sqcup C_2$

$\neg C_1$

$\{a\}$

$\exists P.C$

$\exists P.\mathrm{Self}$

$\forall P.C$

$\forall P.\mathrm{Self}$

$\geqslant P.C$

$\leqslant P.C$

PROPERTIES

$P_1 \circ P_2$

P^-

AXIOMS

$P_1 \sqsubseteq P_2$

$P_1 \equiv P_2$

$\mathrm{disjoint}(P_1, P_2)$

$C_1 \sqsubseteq C_2$

$C_1 \equiv C_2$

$a : C$

$\langle a, b \rangle : R$

$a_1 \approx a_2$

$a_1 \not\approx a_2$

REASONING

$\sqcap$ rule:

$\sqcup$ rule:

$\exists$ rule:

$\forall$ rule:

A-box : our triples

Example

$A = \{ a:C_1 , b:C_2 \}$

an A-box with 2 triples

REASONING

$\sqcap$ rule: $a : (C_1 \sqcap C_2) \in A$

what else goes in the A-box?

$\sqcup$ rule:

$\exists$ rule:

$\forall$ rule:

REASONING

$\sqcap$ rule: if 1. $a : (C_1 \sqcap C_2) \in A$ and

2. $\{a : C_1, a : C_2\} \notin A$

$\sqcup$ rule: then ...

$\exists$ rule:

$\forall$ rule:

REASONING

$\sqcap$ rule: if 1. $a : (C_1 \sqcap C_2) \in A$ and

2. $\{a : C_1, a : C_2\} \notin A$

$\sqcup$ rule: then set $A_1 = A \cup \{a : C_1, a : C_2\}$

$\exists$ rule: $A = \{ a : \text{Engineer} \sqcap \text{Artist} \}$

$A_1 =$

$\forall$ rule:

REASONING

$\sqcap$ rule: if 1. $a : (C_1 \sqcap C_2) \in A$ and

2. $\{a : C_1, a : C_2\} \notin A$

$\sqcup$ rule: then set $A_1 = A \cup \{a : C_1, a : C_2\}$

$\exists$ rule:

Example

$A = \{ a : \text{Engineer} \sqcap \text{Artist} \}$

$\forall$ rule:

$$A_1 = \left\{ \begin{array}{l} a : \text{Engineer}, a : \text{Artist}, \\ a : \text{Engineer} \sqcap \text{Artist} \end{array} \right\}$$

REASONING

$\sqcap$ rule: if 1. $a:(C_1 \sqcap C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \not\subseteq A$
then set $A_1 = A \cup \{a:C_1, a:C_2\}$

$\sqcup$ rule: if 1. $a:(C_1 \sqcup C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \cap A = \phi$
then set

$\exists$ rule:

$\forall$ rule:

REASONING

$\sqcap$ rule: if 1. $a:(C_1 \sqcap C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \not\subseteq A$
then set $A_1 = A \cup \{a:C_1, a:C_2\}$

$\sqcup$ rule: if 1. $a:(C_1 \sqcup C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \cap A = \phi$
then set
$A_1 = A \cup \{a:C_1\}$ and
$A_2 = A \cup \{a:C_2\}$
we branch

$\exists$ rule:

$\forall$ rule:

REASONING

$\sqcap$ rule: if 1. $a:(C_1 \sqcap C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \not\subseteq A$
then set $A_1 = A \cup \{a:C_1, a:C_2\}$

$\sqcup$ rule: if 1. $a:(C_1 \sqcup C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \cap A = \phi$
then set
$A_1 = A \cup \{a:C_1\}$ and
$A_2 = A \cup \{a:C_2\}$
we branch

$\exists$ rule:

$\forall$ rule:

Example

$A = \{a: \text{Engineer} \sqcup \text{Artist}\}$ then

$A_1 = \{a: \text{Engineer} \sqcup \text{Artist}, a: \text{Engineer}\}$

$A_2 = \{a: \text{Engineer} \sqcup \text{Artist}, a: \text{Artist}\}$

REASONING

$\sqcap$ rule: if 1. $a:(C_1 \sqcap C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \not\subseteq A$
then set $A_1 = A \cup \{a:C_1, a:C_2\}$

$\sqcup$ rule: if 1. $a:(C_1 \sqcup C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \cap A = \phi$
then set
$A_1 = A \cup \{a:C_1\}$ and
$A_2 = A \cup \{a:C_2\}$
we branch

$\exists$ rule: if 1. $a:(\exists P.C) \in A$ and
2. there is no b such that
$\{\langle a,b\rangle:P, b:C\} \subseteq A$

$\forall$ rule:
then set

REASONING

$\sqcap$ rule: if 1. $a:(C_1 \sqcap C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \not\subseteq A$

then set $A_1 = A \cup \{a:C_1, a:C_2\}$

$\sqcup$ rule: if 1. $a:(C_1 \sqcup C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \cap A = \phi$

then set
$A_1 = A \cup \{a:C_1\}$ and
$A_2 = A \cup \{a:C_2\}$

$\exists$ rule: if 1. $a:(\exists P.C) \in A$ and

2. there is no b such that

$\{\langle a,b\rangle : P, b:C\} \subseteq A$

then set

$A_1 = A \cup \{\langle a,d\rangle : P, d:C\}$

where d is new in A

$\forall$ rule:

REASONING

$\sqcap$ rule: if 1. $a:(C_1 \sqcap C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \not\subseteq A$

then set $A_1 = A \cup \{a:C_1, a:C_2\}$

$\sqcup$ rule: if 1. $a:(C_1 \sqcup C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \cap A = \phi$

then set
$A_1 = A \cup \{a:C_1\}$ and
$A_2 = A \cup \{a:C_2\}$

$\exists$ rule: if 1. $a:(\exists P.C) \in A$ and
2. there is no b such that
$\{\langle a,b\rangle : P, b:C\} \subseteq A$

then set
$A_1 = A \cup \{\langle a,d\rangle : P, d:C\}$
where d is new in A

Example

$A = \{ a:\exists \text{parent}.\text{Doctor}, b:\text{Doctor}\}$

$A_1 = \{ a:\exists \text{parent}.\text{Doctor}, b:\text{Doctor}, \langle a,d\rangle:\text{parent}, d:\text{Doctor}\}$

$\forall$ rule:

REASONING

$\sqcap$ rule: if 1. $a:(C_1 \sqcap C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \not\subseteq A$

then set $A_1 = A \cup \{a:C_1, a:C_2\}$

$\sqcup$ rule: if 1. $a:(C_1 \sqcup C_2) \in A$ and
2. $\{a:C_1, a:C_2\} \cap A = \phi$

then set
$A_1 = A \cup \{a:C_1\}$ and
$A_2 = A \cup \{a:C_2\}$

$\exists$ rule: if 1. $a:(\exists P.C) \in A$ and
2. there is no b such that
$\{\langle a,b\rangle:P, b:C\} \subseteq A$

then set
$A_1 = A \cup \{\langle a,d\rangle:P, d:C\}$
where d is new in A

$\forall$ rule: if 1. $\{a:\forall P.C, \langle a,b\rangle:P\} \subseteq A$ and
2. $b:C \notin A$

then $A_1 = A \cup \{b:C\}$

Example $A = \{a:\forall hasChild.Doctor, \langle a,b\rangle:hasChild\}$

$A_1 = \{a:\forall hasChild.Doctor, \langle a,b\rangle:hasChild, b:Doctor\}$

REASONING

$\sqcap$ rule: if 1. $a : (C_1 \sqcap C_2) \in A$ and
2. $\{a : C_1, a : C_2\} \not\subseteq A$

then set $A_1 = A \cup \{a : C_1, a : C_2\}$

$\sqcup$ rule: if 1. $a : (C_1 \sqcup C_2) \in A$ and
2. $\{a : C_1, a : C_2\} \cap A = \phi$

then set
$A_1 = A \cup \{a : C_1\}$ and
$A_2 = A \cup \{a : C_2\}$

$\exists$ rule: if 1. $a : (\exists P.C) \in A$ and
2. there is no b such that
$\{\langle a,b\rangle : P, b : C\} \subseteq A$

then set
$A_1 = A \cup \{\langle a,d\rangle : P, d : C\}$
where d is new in A

$\forall$ rule: if 1. $\{a : \forall P.C, \langle a,b\rangle : P\} \subseteq A$ and
2. $b : C \notin A$

then $A_1 = A \cup \{b : C\}$

CLASSES

$\top$

$\bot$

$C_1 \sqcap C_2$

$C_1 \sqcup C_2$

$\neg C_1$

$\{a\}$

$\exists P.C$

$\exists P.\mathrm{Self}$

$\forall P.C$

$\forall P.\mathrm{Self}$

$\geqslant P.C$

$\leqslant P.C$

PROPERTIES

$P_1 \circ P_2$

P^-

AXIOMS

$P_1 \sqsubseteq P_2$

$P_1 \equiv P_2$

$\mathrm{disjoint}(P_1, P_2)$

$C_1 \sqsubseteq C_2$

$C_1 \equiv C_2$

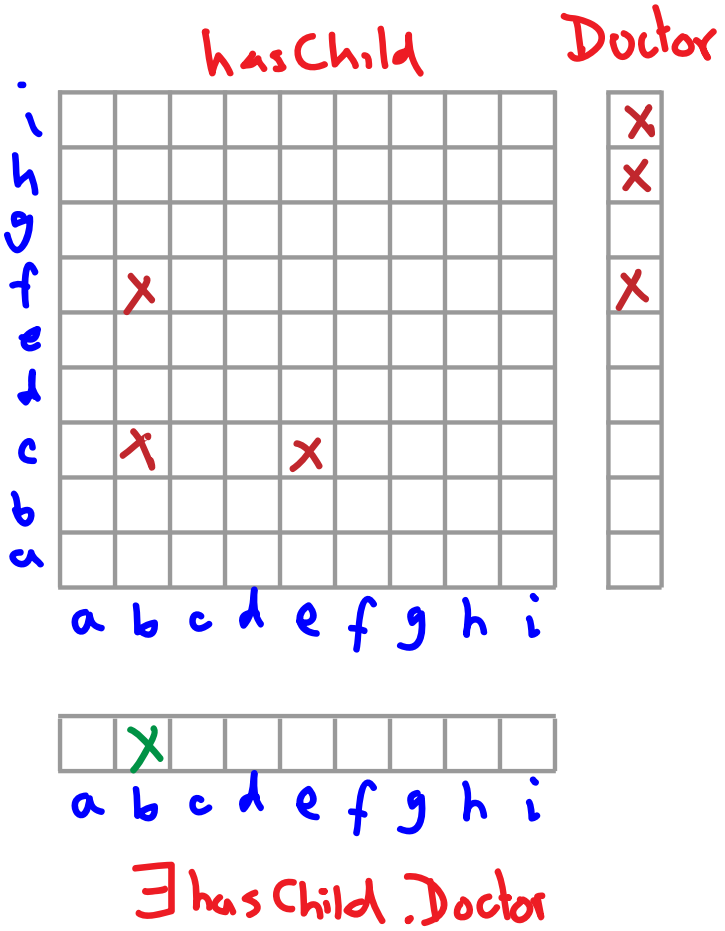
$a : C$

$\langle a, b \rangle : P$

$a_1 \approx a_2$

$a_1 \not\approx a_2$

∃ P . C



set of triples $\{x_i\}$

such that

$\langle x_i, y_i \rangle : P \wedge y_i : C$

individuals who

have a child

who is a Doctor

$\exists P.C$

$\exists$ hasChild . Doctor

individuals who
have a child who is a Doctor

$\exists$ hasChild . Doctor $\sqcap$ Person

persons who
have a child who is a Doctor

$\exists$ hasChild . Doctor $\sqcap$ Person $\sqsubseteq \bot$ meaning?

$\exists$ hasChild . Doctor $\sqcap$ Person $\sqsupseteq \top$

$\forall$ hasChild . Doctor $\sqcap$ Person

} different?

EXAMPLE

$\mathrm{Doctor} \sqsubseteq \mathrm{Person}$

$\mathrm{Parent} \equiv \mathrm{Person} \sqcap \exists \mathrm{hasChild}.\mathrm{Person}$

$\mathrm{HappyParent} \equiv \mathrm{Parent} \sqcap \forall \mathrm{hasChild}.(\mathrm{Doctor} \sqcup \exists \mathrm{hasChild}.\mathrm{Doctor})$

translate to English

EXAMPLE

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$A =$ John : HappyParent,
$\langle John, Mary \rangle$: hasChild

is Mary : Doctor ?

is Mary : Doctor ?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$\mathcal{A} =$ $John : HappyParent$,
$\langle John, Mary \rangle : hasChild$
$Mary : \neg Doctor$ add the negation, and apply the rules

is Mary : Doctor ?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$A =$ $John : HappyParent$,

$\langle John, Mary \rangle : hasChild$

$Mary : \neg Doctor$

$John : Parent$

$John : \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : Person$

$John : \exists hasChild.Person$

is Mary:Doctor?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : HappyParent$,

$\langle John, Mary \rangle : hasChild$

$Mary : \neg Doctor$

$John : Parent$

$John : \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : Person$

$John : \exists hasChild.Person$

$Mary : Doctor$

$Mary : \exists hasChild.Doctor$

is Mary : Doctor ?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : HappyParent$,

$\langle John, Mary \rangle : hasChild$

$Mary : \neg Doctor$ inconsistent

$John : Parent$

$John : \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : Person$

$John : \exists hasChild.Person$

~~$Mary : Doctor$~~

$Mary : \exists hasChild.Doctor$

is Mary:Doctor?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : HappyParent$,

$\langle John, Mary \rangle : hasChild$

$Mary : \neg Doctor$

$John : Parent$

$John : \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : Person$

$John : \exists hasChild.Person$

$Mary : \exists hasChild.Doctor$

$\langle John, a \rangle : hasChild$

$a : Doctor \sqcup \exists hasChild.Doctor$

is Mary:Doctor?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : HappyParent$,

$\langle John, Mary \rangle : hasChild$

$Mary : \neg Doctor$

$John : Parent$

$John : \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : Person$

$John : \exists hasChild.Person$

$Mary : \exists hasChild.Doctor$

$\langle John, a \rangle : hasChild$

$a : Doctor \sqcup \exists hasChild.Doctor$

$\langle Mary, b \rangle : hasChild$

$b : Doctor$

More inferences

$a : Person$

$b : Person$

is Mary:Doctor?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : HappyParent$,

$\langle John, Mary \rangle : hasChild$

$Mary : \neg Doctor$

$John : Parent$

$John : \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : Person$

$John : \exists hasChild.Person$

$Mary : \exists hasChild.Doctor$

$\langle John, a \rangle : hasChild$

$a : Doctor \sqcup \exists hasChild.Doctor$

$\langle Mary, b \rangle : hasChild$

$b : Doctor$

$a : Person$

$b : Person$

We are stuck, and we didn't reach a contradiction!

we cannot infer Mary:Doctor

is Mary : Doctor ?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : HappyParent$,

$\langle John, Mary \rangle : hasChild$

Suppose we add one more axiom

$Mary : \forall hasChild.\bot$

$Mary : \neg Doctor$

$John : Parent$

$John : \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : Person$

$John : \exists hasChild.Person$

$Mary : \exists hasChild.Doctor$

$\langle John, a \rangle : hasChild$

$a : Doctor \sqcup \exists hasChild.Doctor$

$\langle Mary, b \rangle : hasChild$

$b : Doctor$

$a : Person$

$b : Person$

is Mary : Doctor ?

$Doctor \sqsubseteq Person$

$Parent \equiv Person \sqcap \exists hasChild.Person$

$HappyParent \equiv Parent \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : HappyParent$,

$\langle John, Mary \rangle : hasChild$

Suppose we add one more axiom

$Mary : \forall hasChild.\bot$

$Mary : \neg Doctor$

$John : Parent$

$John : \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$

$John : Person$

$John : \exists hasChild.Person$

$Mary : \exists hasChild.Doctor$

$\langle John, a \rangle : hasChild$

$a : Doctor \sqcup \exists hasChild.Doctor$

$\langle Mary, b \rangle : hasChild$

$b : Doctor$

$b : \bot$ YEAH! Contradiction!

$a : Person$

$b : Person$

PROFILES

SROIQ exponential

OWL-EL polynomial

OWL-RL polynomial

OWL-QL aka DL-Lite Logspace, same as SQL

SROIQ

$\top$

$\bot$

$\exists P.C$

$\exists P.\mathrm{Self}$

~~$P_1 \circ P_2$~~

$P_1 \circ P_2 \sqsubseteq P_3$

$P_1 \sqsubseteq P_2$

$P_1 \equiv P_2$

$\mathrm{disjoint}(P_1, P_2)$

$C_1 \sqcap C_2$

$C_1 \sqcup C_2$

$\neg C_1$

$\forall P.C$

$\forall P.\mathrm{Self}$

P^-

$C_1 \sqsubseteq C_2$

$C_1 \equiv C_2$

$\{a\}$

$\geqslant P.C$

$\leqslant P.C$

$a:C$

$\langle a,b \rangle : P$

$a_1 \approx a_2$

$a_1 \not\approx a_2$

decidable, but exponential

OWL-EL

$\top$

$\bot$

$\exists P.C$

$\exists P.\mathrm{Self}$

~~$P_1 \circ P_2$~~

$P_1 \circ P_2 \sqsubseteq P_3$

restricted

$P_1 \sqsubseteq P_2$

$P_1 \equiv P_2$

$\mathrm{disjoint}(P_1, P_2)$

$C_1 \sqcap C_2$

~~$C_1 \sqcup C_2$~~

~~$\neg C_1$~~

$\forall P.C$

$\forall P.\mathrm{Self}$

P^-

$C_1 \sqsubseteq C_2$

$C_1 \equiv C_2$

$\{a\}$

~~$\geqslant P.C$~~

~~$\leqslant P.C$~~

$a:C$

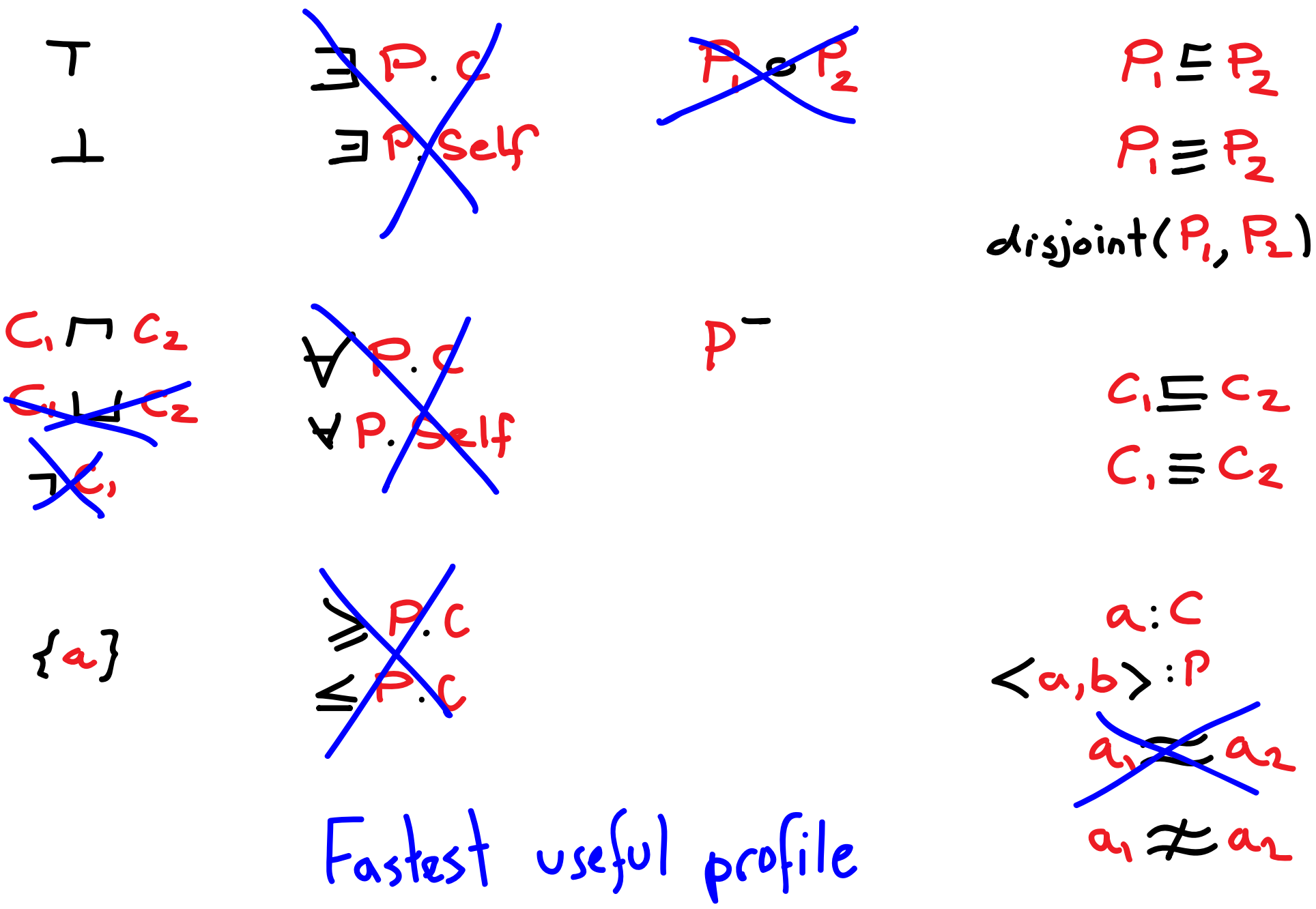
$\langle a,b \rangle : P$

$a_1 \approx a_2$

$a_1 \not\approx a_2$

polynomial

OWL-QL



Fastest useful profile

OWL-QL

$\top$ ✳

$C_1 \sqcap C_2$

P^-

domain(P, C)

range(P, C)

$P_1 \sqsubseteq P_2$

$P_1 \equiv P_2$

$C_1 \sqsubseteq C_2$

$C_1 \equiv C_2$

$a : C$

$\langle a, b \rangle : P$

$a_1 \neq a_2$

disjoint(C_1, C_2)

disjoint(P_1, P_2)

symmetric(P)

OWL Profile Validator

http://mowl-power.cs.man.ac.uk:8080/validator/