

# Semantics of Notation3 Logic: From implicit to explicit quantification

Dörthe Arndt, September 2018

# Outline

Notation3 Logic

Implicit Quantification

Core Logic

Mapping

# Outline

Notation3 Logic

Implicit Quantification

Core Logic

Mapping

# What is Notation3 Logic?

- Rule-based logic for the Semantic Web
- Invented by Tim Berners-Lee and Dan Connolly (~2005)
- Superset of RDF/Turtle
- There are different reasoners, among them CWM and EYE

# Syntax

Simple triples of subject, predicate, object

`:Kurt :knows :Albert.` -> *“Kurt knows Albert”*

Conjunction of triples by simply listing them together:

`:Kurt :knows :Albert. Albert :knows :Kurt.`

-> *“Kurt knows Albert **and** Albert knows Kurt.”*

# Syntax

## Citation of formulas

`:John :says { :Kurt :knows :Albert. } .`

-> “*John says **that** Kurt knows Albert.*”

## Logical Implication

`{ :Kurt :knows :Albert. } => { :Albert :knows :Kurt. } .`

-> “*If Kurt knows Albert **then** Albert knows Kurt.*”

# Outline

Notation3 Logic

**Implicit Quantification**

Core Logic

Mapping

# What is implicit quantification?

In N3 quantified variables can be expressed without explicitly stating the quantifier

- > Blank nodes `_:x` are **existentially quantified** variables
- > Variables beginning with a question mark `?x` are **universally quantified**

But: What is the scope? What is the position of the quantifier?



# Implicit Quantification: Existentials

RDF's blank nodes are interpreted as existentially quantified variables:

`_ :x :knows :Albert.`

->  $\exists x: \text{knows}(x, \text{Albert})$

***“There exists someone who knows Albert.”***

# Implicit Quantification: Universals

N3 has a notation for implicitly quantified variables:

`{ ?x : knows :Albert } => { :Albert : knows ?x } .`

->  $\forall x: \text{knows}(x, \text{Albert}) \rightarrow \text{knows}(\text{Albert}, x)$

***“Everyone who knows Albert is also known by Albert.”***

# Combination of quantifiers

`_ :x :thinks {?y :is :pretty} .`

?

~~->  $\exists x \forall y$  thinks(x, is(y, pretty))~~

~~*“There exists someone who thinks  
that everyone is pretty.”*~~

->  $\forall y \exists x$  thinks(x, is(y, pretty))

*“There exists someone who thinks  
that everyone is pretty.”*



# Nested existentials

$\_ : x : \text{says } \{ \_ : x : \text{knows } : \text{Albert} \} .$

?

~~$\exists x_1 : \text{says}(x_1, (\exists x_2 : \text{knows}(x_2, \text{Albert})))$~~

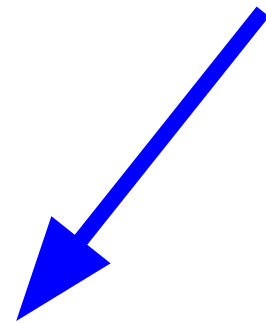
~~*“There exists someone who says that someone (possibly someone else) knows Albert.”*~~

$\exists x : \text{says}(x, (\text{knows}(x, \text{Albert})))$

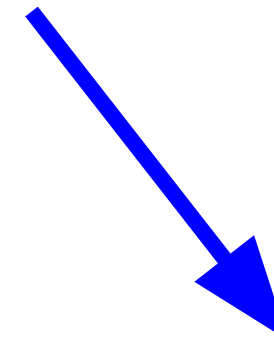
*“There exists someone who says about himself that he knows Albert.”*



# Nested universals

$$\{ \{ ?\mathbf{x} : q : b . \} \Rightarrow \{ : a : r : c . \} . \} \Rightarrow \{ : s : p : o \} .$$


?



$\forall \mathbf{x}: ((q(x,b) \rightarrow r(a, c)) \rightarrow p(s,o))$

$(\forall \mathbf{x}: (q(x,b) \rightarrow r(a, c)) \rightarrow p(s,o))$



EYE



n3

CWM

# Cwm's Interpretation

Universals are quantified on their parent formula:

↓  
 $\{ \{ ?x : q : b . \} \Rightarrow \{ : a : r : c . \} . \} \Rightarrow \{ : s : p : o \} .$

direct formula

parent formula

$\rightarrow (\forall x : q(x, b) \rightarrow r(a, c)) \rightarrow p(s, o)$

# EYE's Interpretation

Universals are always quantified on top level:

↓  
 $\{ \{ ?\mathbf{x} : q : b . \} \Rightarrow \{ :a : r : c . \} . \} \Rightarrow \{ :s : p : o \} .$

→  $\forall \mathbf{x} : (q(x, b) \rightarrow r(a, c)) \rightarrow p(s, o)$

# Problem

In the examples we used natural language and a first order logic like syntax

**But:** for these two languages the semantics is not clearly defined!

-> We need a way to express these differences in a logic with well-defined  
Semantics!



# Outline

Notation3 Logic

Implicit Quantification

**Core Logic**

Mapping

# Core Logic

We define a simple logic similar to N3 with the only difference that **only explicit quantification** is allowed.

# Overview: Syntax

Similar to N3, but:

no distinction between  
universal and existential  
variables

explicit quantification

{ and } become < and >

$t ::=$	$v$ $c$ $e$ $(k)$ $()$	terms: variables constants expressions lists empty list
$k ::=$	$t$ $t\ k$	listcontent
$e ::=$	$\langle f \rangle$ $\langle \rangle$ $false$	expressions: formula expression true false
$f ::=$	$t\ t\ t$ $e \rightarrow e$ $f\ f$ $\forall v.f$ $\exists v.f$	formulas: atomic formula implication conjunction universal formula existential formula

# Semantics

For **ground terms** (= terms without variables) we define **two mappings** into the domain of discourse:

## Definition (Structure)

A structure over a language  $\mathcal{L}$  is a triple  $\mathfrak{A} = (\mathcal{D}, \alpha, \mathfrak{p})$  containing:

- ▶ A non empty set  $\mathcal{D}$  called the **domain**.
- ▶ A mapping  $\alpha : \mathcal{T}_g \rightarrow \mathcal{D}$  called the **object mapping**.
- ▶ A mapping  $\mathfrak{p} : \mathcal{T}_g \rightarrow 2^{\mathcal{D} \times \mathcal{D}}$  called the **predicate mapping**.

# Semantics

Why **two mappings**?

-> N3 allows quantification over variables in predicate position

$$\{ ?s \text{ } \textbf{?p} \text{ } ?o. \text{ } \textbf{?p} \text{ } \texttt{rdfs:subPropertyOf} \text{ } \textbf{?q}. \} \Rightarrow \{ ?s \text{ } \textbf{?q} \text{ } ?o. \} .$$

# Semantics

What about non-ground terms?

- Non-ground terms get grounded via a grounding function  $\gamma$
- An interpretation  $\mathcal{I}$  consists of a structure  $\mathcal{A}$  and a grounding function  $\gamma$

# Semantics

## Definition (Meaning of formulas)

Let  $\mathcal{I} = (\mathcal{A}, \gamma)$  be an interpretation of a language  $\mathcal{L}$ . Then:

1.  $\mathcal{I} \models t_1 t_2 t_3$  iff  $(a(\gamma(t_1)), a(\gamma(t_3))) \in p(\gamma(t_2))$
2.  $\mathcal{I} \models \langle f_1 \rangle \rightarrow \langle f_2 \rangle$  iff  $\mathcal{I} \models f_2$  if  $\mathcal{I} \models f_1$ .
3.  $\mathcal{I} \models \text{false} \rightarrow \langle f \rangle$
4.  $\mathcal{I} \models \langle f \rangle \rightarrow \langle \rangle$
5.  $\mathcal{I} \models \langle f \rangle \rightarrow \text{false}$  iff  $\mathcal{I} \not\models f$ .
6.  $\mathcal{I} \models \langle \rangle \rightarrow \langle f \rangle$  iff  $\mathcal{I} \models f$ .
7.  $\mathcal{I} \models f_1 f_2$  iff  $\mathcal{I} \models f_1$  and  $\mathcal{I} \models f_2$ .
8.  $\mathcal{I} \models \forall v. f$  iff  $(\mathcal{A}, \gamma[v \mapsto t]) \models f$  for all  $t \in \mathcal{T}_g$ .
9.  $\mathcal{I} \models \exists v. f$  iff  $(\mathcal{A}, \gamma[v \mapsto t]) \models f$  for some  $t \in \mathcal{T}_g$ .

We call a formula  $f$  *true* in  $\mathcal{I}$  if  $\mathcal{I} \models f$ .

# Semantics

Why do we **ground first**?

- To not quantify over the powerset of the domain of discourse
- To be able to support **built-in functions** which deal on the representation level of the logic (e.g. `log:equalTo`) as well as concepts which are defined on domain level (e.g. `owl:sameAs`)



# Semantics

(At least) two kinds of equality:

- **Equality on domain level:**

`dbpedia:Bern owl:sameAs dbpedia-nl:Bern. -> true`

- **Equality on representation level:**

`dbpedia:Bern log:equalTo dbpedia-nl:Bern. -> false`

`dbpedia:Bern log:equalTo dbpedia:Bern. -> true`

*Remark: The unique name assumption makes these two equalities the same, but UNA cannot be assumed in the Semantic Web*

# Outline

Notation3 Logic

Implicit Quantification

Core Logic

Mapping

# Mapping

Back to our original problem:

How do we translate the different interpretations into Core Logic?

- For EYE, this translation is simple as universals are quantified on top level, existentials in the formula they occur in
- For CWM we make use of the **syntax tree** of a formula

# Syntax of N3

Quantifiers in the Core Logic  
can occur at two kinds of  
nodes:

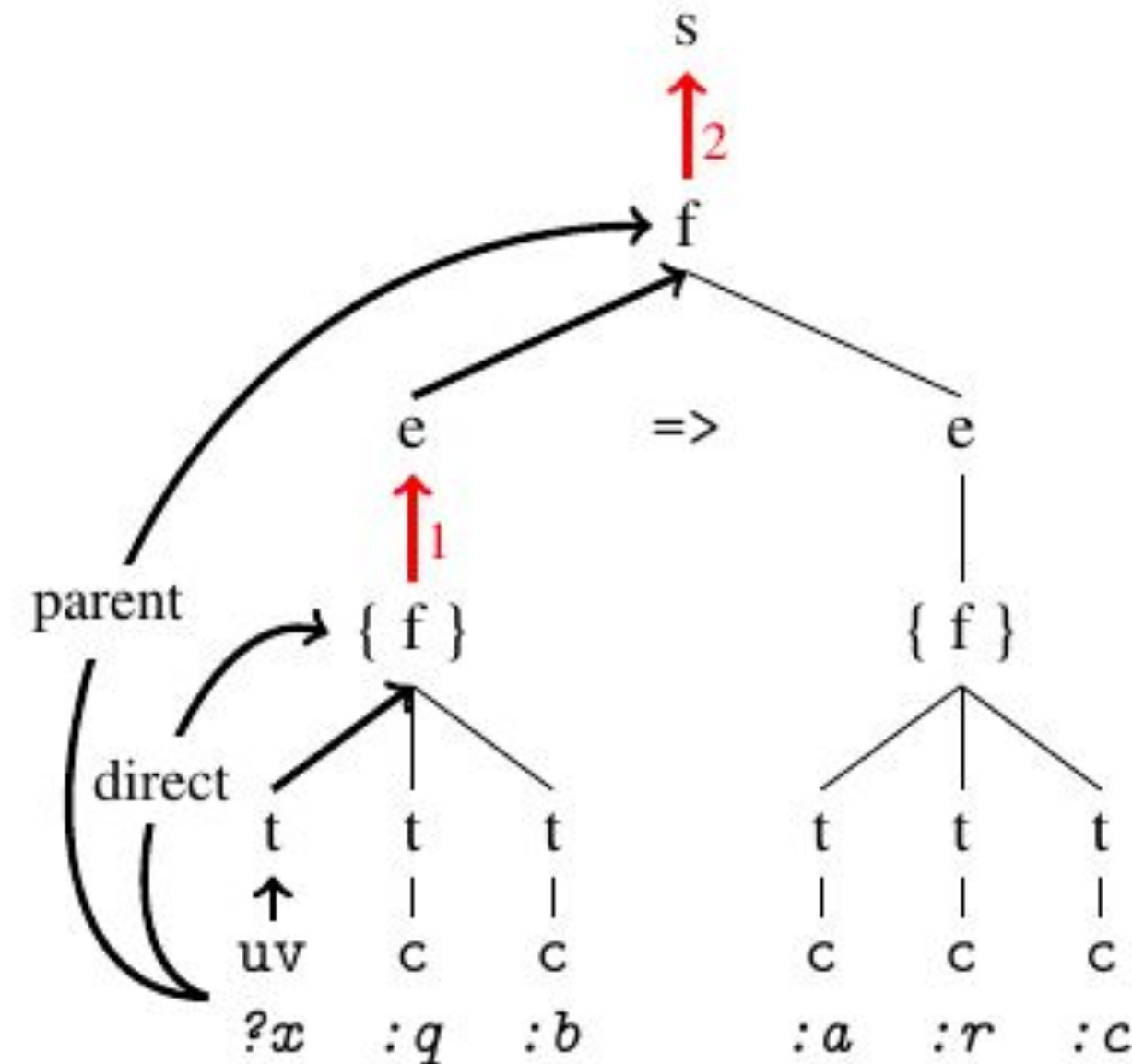
- the start node  $s$
- a node  $e$  which produces  
a new expression ( $\approx$   
formula in curly brackets)

$s ::=$	$f$	start: formula
$f ::=$	$t \ t \ t.$ $e \Rightarrow e.$ $f \ f$	formulas: atomic formula implication conjunction
$t ::=$	$uv$ $ex$ $c$ $e$ $(k)$ $()$	terms: universal variables existential variables constants expressions lists empty list
$k ::=$	$t$ $t \ k$	list content: term term tail
$e ::=$	$\{f\}$ $\{\}$ $false$	expressions: formula expression true false

# How can syntax trees help?

The parent of a universal variable can be defined using the means of the syntax tree:

Going from the bottom to the top of the tree, the parent formula is the **second** formula **f** which is **child of** either a node **e** or the start node **s**



# How can we formalise that idea?

We use an **attribute grammar**

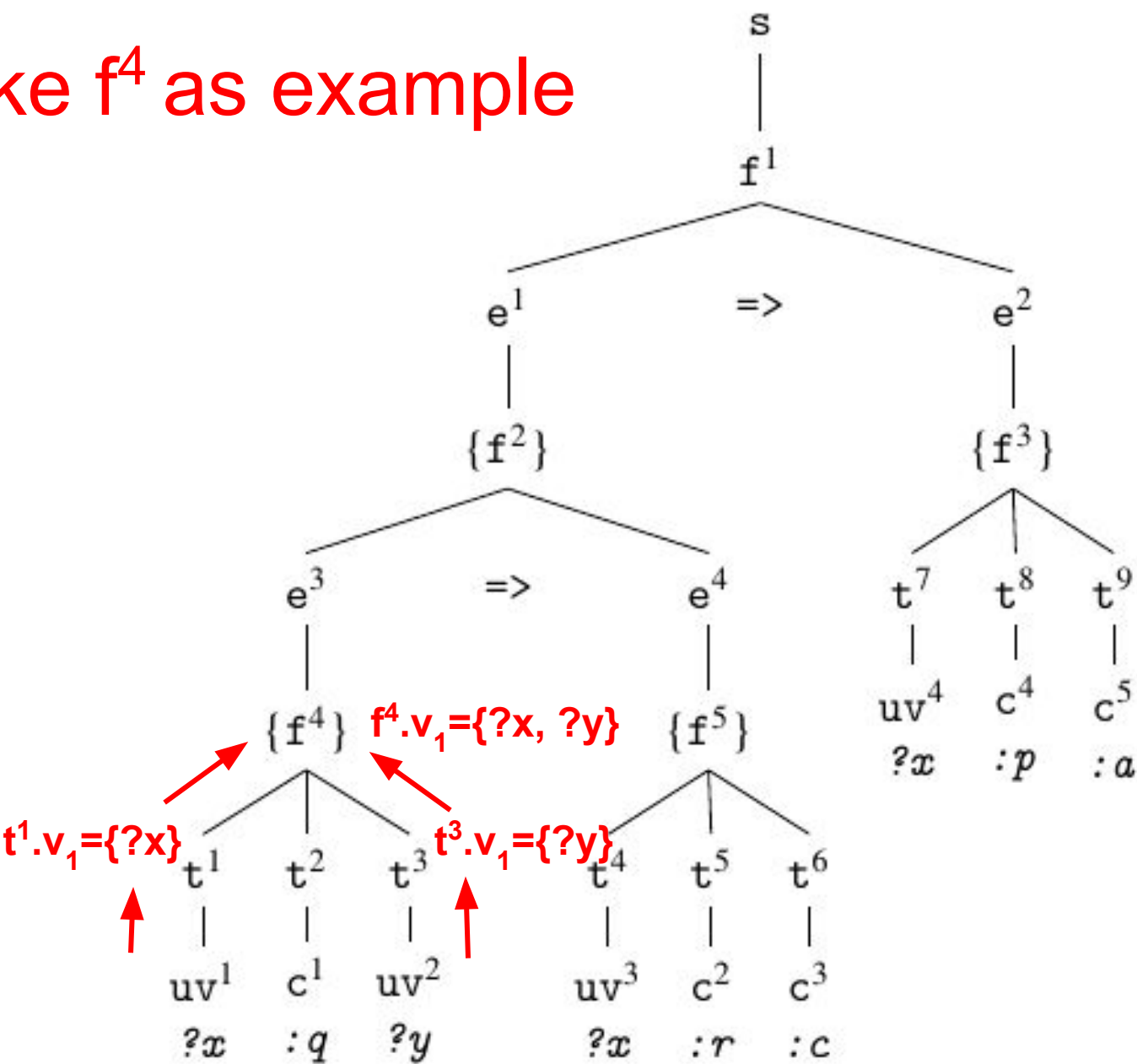
- attribute grammars provide a mechanism to pass information through a syntax tree of a context-free grammar
- to do so, **attributes** are defined on top of the production rules of the CFG
- attributes can either be **inherited** or **synthesized**
- **inherited attributes** pass information **downwards** in the syntax tree
- **synthesized attributes** pass information **upwards** in the syntax tree

# Example

To pass universals to their direct formula, define the synthesized attribute  $v_1$ :

production rules	rules for $v_1$
$s ::= f$	$s.v_1 \leftarrow \emptyset$
$f ::= t_1 t_2 t_3.$	$f.v_1 \leftarrow t_1.v_1 \cup t_2.v_1 \cup t_3.v_1$
$e_1 \Rightarrow e_2.$	$f.v_1 \leftarrow e_1.v_1 \cup e_2.v_1$
$f_1 f_2$	$f.v_1 \leftarrow f_1.v_1 \cup f_2.v_1$
$t ::= uv$	$t.v_1 \leftarrow \{uv\}$
$ex$	$t.v_1 \leftarrow \emptyset$
$c$	$t.v_1 \leftarrow \emptyset$
$e$	$t.v_1 \leftarrow e.v_1$
$(k)$	$t.v_1 \leftarrow k.v_1$
$()$	$t.v_1 \leftarrow \emptyset$
$k ::= t$	$k.v_1 \leftarrow t.v_1$
$t\ k_1$	$k.v_1 \leftarrow t.v_1 \cup k_1.v_1$
$e ::= \{f\}$	$e.v_1 \leftarrow \emptyset$
$\{\}$	$e.v_1 \leftarrow \emptyset$
$false$	$e.v_1 \leftarrow \emptyset$

We take  $f^4$  as example



# Attribute Grammar

We furthermore define:

- a **synthesized attribute**  $v_2$  to pass universals from a direct formula to the parent
- an **inherited attribute**  $s$  to pass the information downwards which variables are already quantified under a formula
- an **inherited attribute**  $q$  to determine the exact set of universals which need to be quantified on a formula



# Attribute Grammar

Why do we need the attribute  $s$ ?

$\{\{?x :q \ ?y\} \Rightarrow \{?x :r :c\}\} \Rightarrow \{?x :p :a\}.$

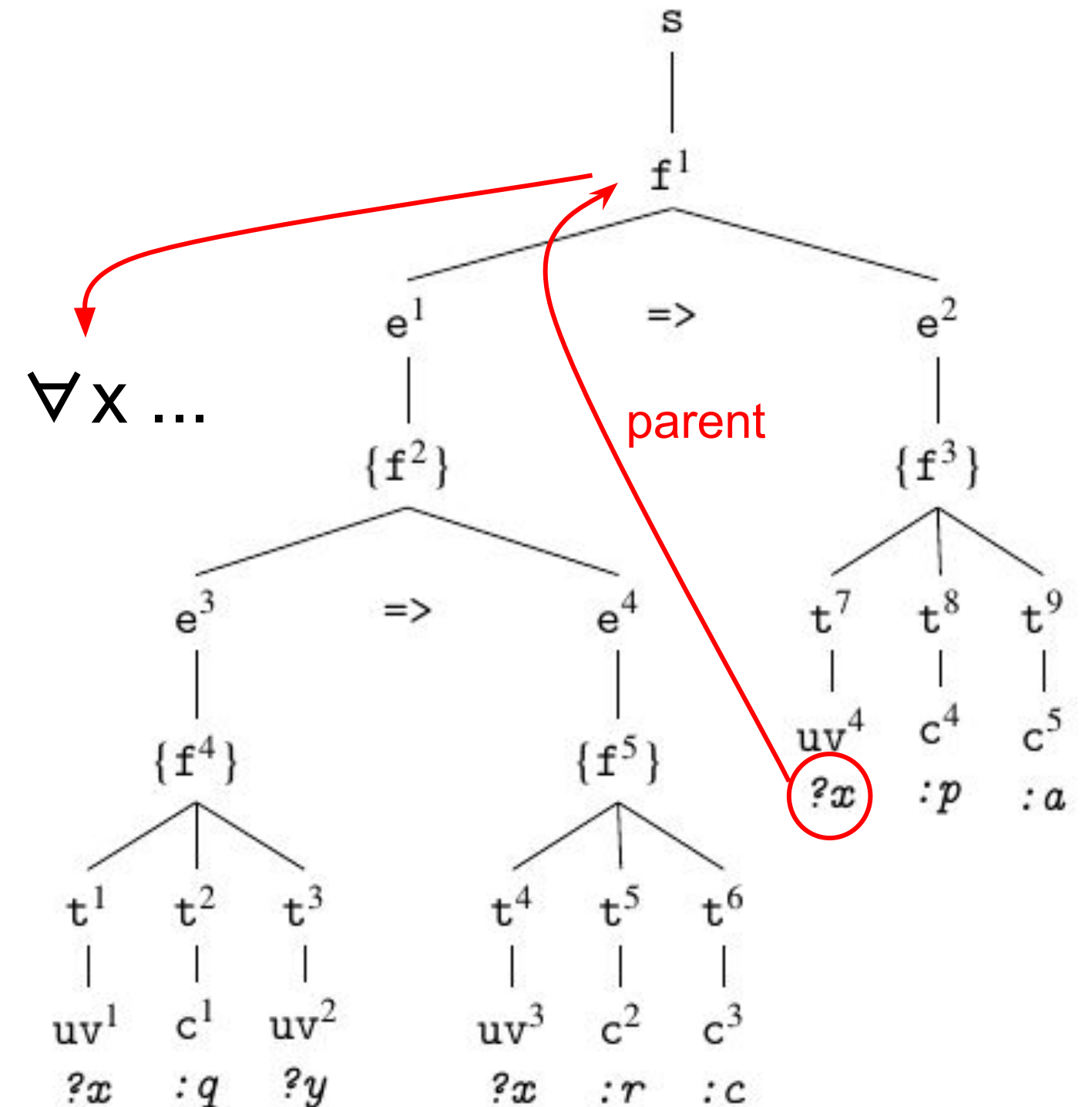
translates to

$\forall x. (\forall y. \langle x \ q \ y \rangle \rightarrow \langle x \ r \ c \rangle) \rightarrow \langle x \ p \ a \rangle$

and **not** to

$\forall x. (\forall x. \forall y. \langle x \ q \ y \rangle \rightarrow \langle x \ r \ c \rangle) \rightarrow \langle x \ p \ a \rangle$

In formula  $f^2$ ,  $?x$  is already scoped.



# Attribute Grammar for universals in CWM

CFG		Synthesized attributes		Inherited attributes	
production rules		rules for $v_1$	rules for $v_2$	rules for $s$	rules for $q$
$s ::=$	$f$	$s.v_1 \leftarrow \emptyset$	$s.v_2 \leftarrow f.v_1$	$f.s \leftarrow f.v_1 \cup f.v_2$	$f.q \leftarrow f.v_1 \cup f.v_2$
$f ::=$	$t_1 t_2 t_3.$	$f.v_1 \leftarrow t_1.v_1 \cup t_2.v_1 \cup t_3.v_1$	$f.v_2 \leftarrow t_1.v_2 \cup t_2.v_2 \cup t_3.v_2$	$t_i.s \leftarrow f.s$	
	$e_1 \Rightarrow e_2.$	$f.v_1 \leftarrow e_1.v_1 \cup e_2.v_1$	$f.v_2 \leftarrow e_1.v_2 \cup e_2.v_2$	$e_i.s \leftarrow f.s$	
	$f_1 f_2$	$f.v_1 \leftarrow f_1.v_1 \cup f_2.v_1$	$f.v_2 \leftarrow f_1.v_2 \cup f_2.v_2$	$f_i.s \leftarrow f.s$	$f_i.q \leftarrow \emptyset$
$t ::=$	$uv$	$t.v_1 \leftarrow \{uv\}$	$t.v_2 \leftarrow \emptyset$		
	$ex$	$t.v_1 \leftarrow \emptyset$	$t.v_2 \leftarrow \emptyset$		
	$c$	$t.v_1 \leftarrow \emptyset$	$t.v_2 \leftarrow \emptyset$		
	$e$	$t.v_1 \leftarrow e.v_1$	$t.v_2 \leftarrow e.v_2$	$e.s \leftarrow t.s$	
	$(k)$	$t.v_1 \leftarrow k.v_1$	$t.v_2 \leftarrow k.v_2$	$k.s \leftarrow t.s$	
	$()$	$t.v_1 \leftarrow \emptyset$	$t.v_2 \leftarrow \emptyset$		
$k ::=$	$t$	$k.v_1 \leftarrow t.v_1$	$k.v_2 \leftarrow t.v_2$	$t.s \leftarrow k.s$	
	$t \ k_1$	$k.v_1 \leftarrow t.v_1 \cup k_1.v_1$	$k.v_2 \leftarrow t.v_2 \cup k_1.v_2$	$t.s \leftarrow k.s$	
				$k_1.s \leftarrow k.s$	
$e ::=$	$\{f\}$	$e.v_1 \leftarrow \emptyset$	$e.v_2 \leftarrow f.v_1$	$f.s \leftarrow e.s \cup f.v_2$	$f.q \leftarrow f.v_2 \setminus e.s$
	$\{\}$	$e.v_1 \leftarrow \emptyset$	$e.v_2 \leftarrow \emptyset$		
	$false$	$e.v_1 \leftarrow \emptyset$	$e.v_2 \leftarrow \emptyset$		

# Outline

Notation3 Logic

Implicit Quantification

Core Logic

Mapping

# Implementation

We used the attribute grammar discussed above and implemented a system that translates N3 formulas to Core Logic formulas following the interpretations of EYE and Cwm.

The implementation can be used to test whether these reasoners conflict in their interpretation of given formulas.

The implementation is available at:

<https://github.com/IDLabResearch/N3CoreLogic>



**imec**

**IDLab**  
INTERNET & DATA LAB