FACULTY OF ENGINEERING AND ARCHITECTURE

IDLab INTERNET & DATA LAB

Semantics of Notation3 Logic: From implicit to explicit quantification

Dörthe Arndt, September 2018

GHENT UNIVERSITY

imec

Outline

Notation3 Logic

Implicit Quantification

N3 Core Logic

Mapping

Outline

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

Problem

Notation3 Logic is used in practice, but its semantics has not been formally defined (yet)

Reasoners differ in their interpretation of formulas!

One of the problems is **implicit quantification**

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.”*

GHENT UNIVERSITY

imec

Outline

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.
```

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

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

Implicit Quantification: Universals

N3 has a notation for implicitly quantified variables:

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

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

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

Combination of quantifiers

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

?

~~-> ∃x ∀y thinks(x, is(y, pretty))~~

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

-> ∀y ∃x thinks(x, is(y, pretty))

“For everyone there exists someone who thinks that he/she is pretty.” ✓

Nested existentials

```
_:x :says {_:x :knows :Albert}.
```

?

~~∃x: says(x,(knows(x,Albert))~~

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

$\exists x_1$: says(x_1,($\exists x_2$: knows(x_2,Albert))

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

Nested universals

```
{{?x :q :b.}=>{:a :r :c.}.}=>{:s :p :o}.
```

?

->*"Does* `{:k :q :b.}=>{:a :r :c.}.` *entail* `:s :p :o.`*?"*

-> Yes.

-> No.

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

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

EYE

n3 CWM

GHENT UNIVERSITY

imec

Cwm’s Interpretation

Universals are quantified on their parent formula:

$$\underbrace{\overset{\downarrow}{\texttt{\{}}\underbrace{\texttt{\{?x :q :b.\}}}_{\text{direct formula}}\texttt{=>\{:a :r :c.\}.\}}}_{\text{parent formula}}\texttt{=>\{:s :p :o\}.}$$

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

EYE's Interpretation

Universals are always quantified on top level:

```
{{?x :q :b.}=>{:a :r :c.}.}=>{:s :p :o}.
```

```
-> ∀x: (q(x,b)→r(a,c))→p(s,o)
```

GHENT UNIVERSITY imec

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

N3 Core Logic

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

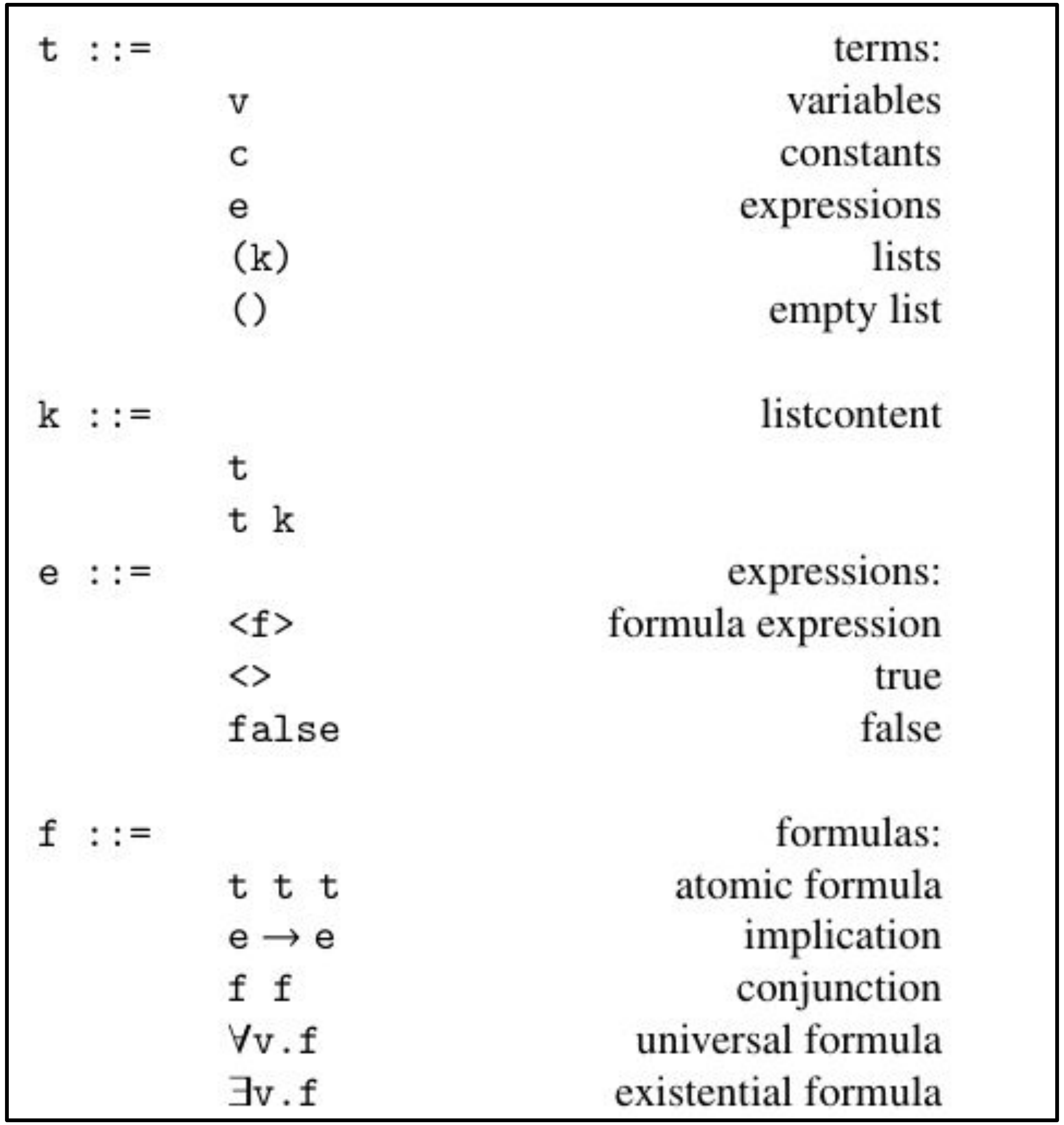
Overview: Syntax

Similar to N3, but:

no distinction between universal and existential variables

explicit quantification

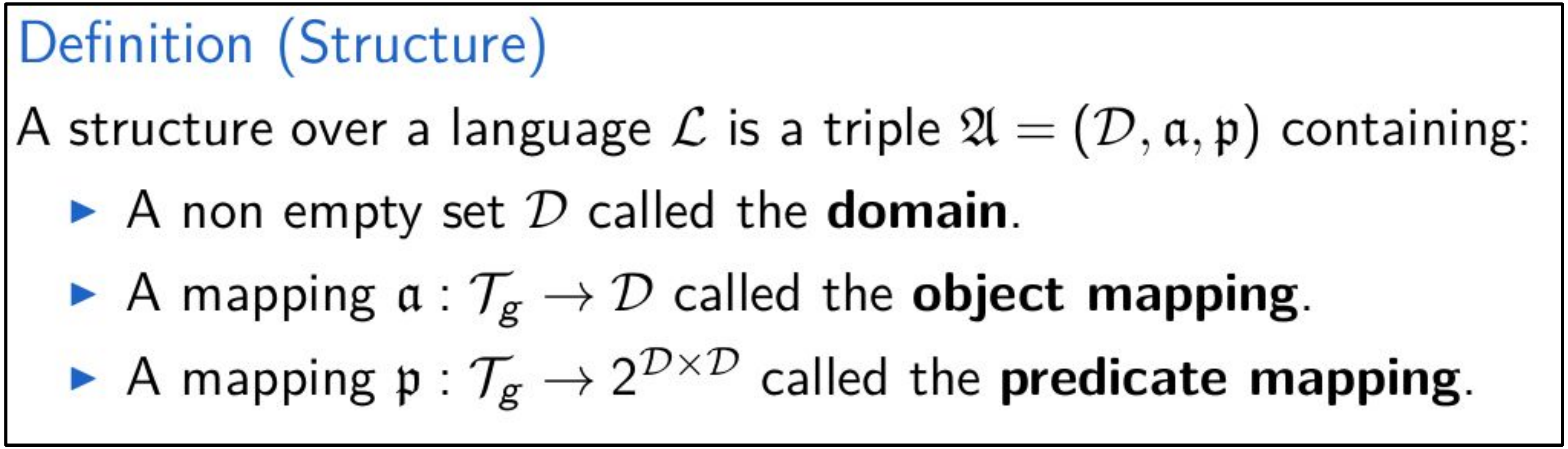
{ and } become < and >



t ::=		terms:
	v	variables
	c	constants
	e	expressions
	(k)	lists
	()	empty list
k ::=		listcontent
	t	
	t k	
e ::=		expressions:
	<f>	formula expression
	<>	true
	false	false
f ::=		formulas:
	t t t	atomic formula
	e → e	implication
	f f	conjunction
	∀v.f	universal formula
	∃v.f	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 $\mathfrak{A} = (\mathcal{D}, \mathfrak{a}, \mathfrak{p})$ containing:

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

Semantics

Why **two mappings**?

-> N3 allows quantification over variables in predicate position

```
{?s ?p ?o. ?p rdfs:subPropertyOf ?q. } => {?s ?q ?o.}.
```

Semantics

What about non-ground terms?

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

Semantics

Definition (Meaning of formulas)

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

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

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

GHENT UNIVERSITY imec

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

GHENT UNIVERSITY imec

Outline

Mapping

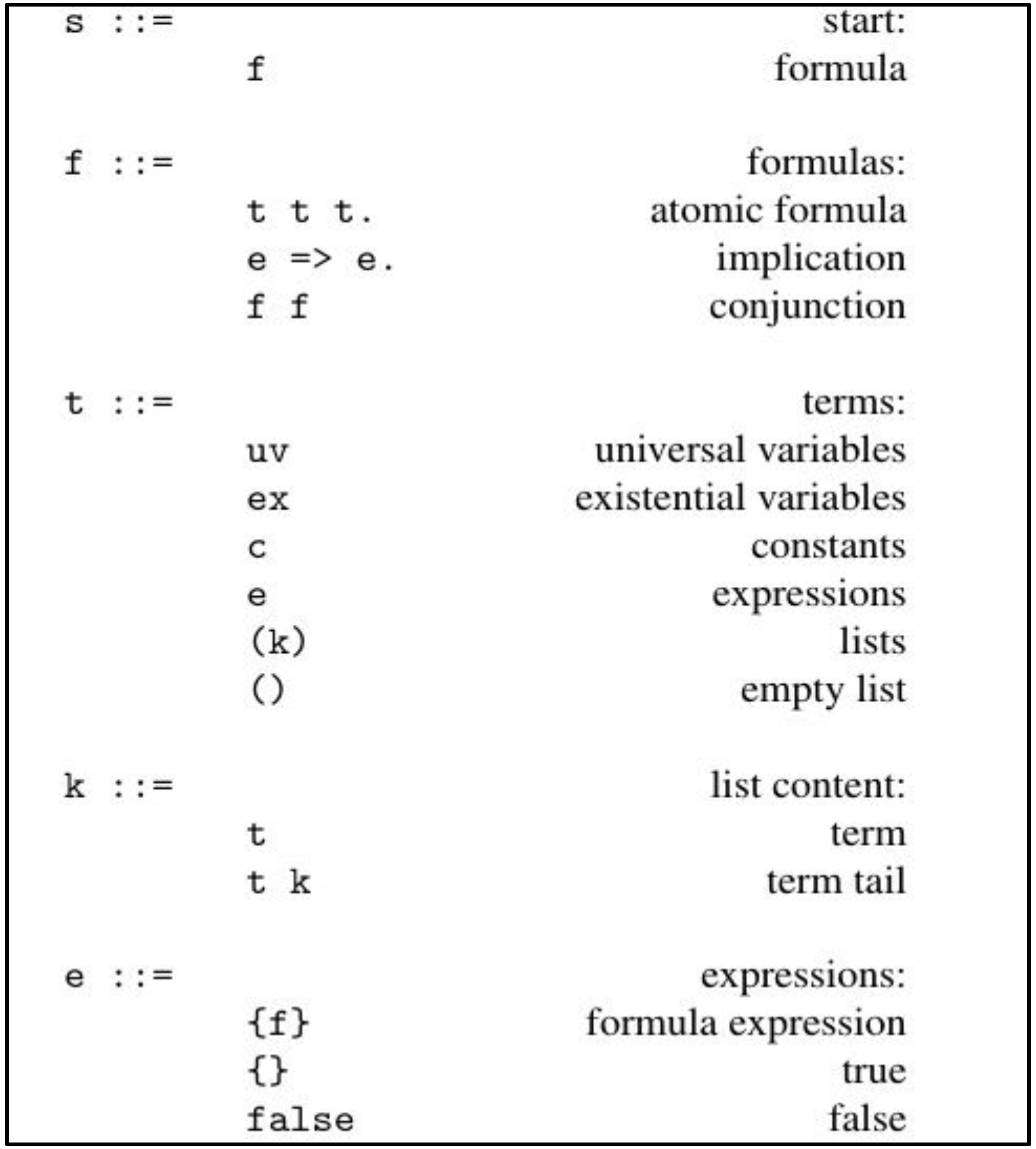
Back to our original problem:
How do we translate the different interpretations into Core Logic?

- 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 (≈ formula in curly braces)

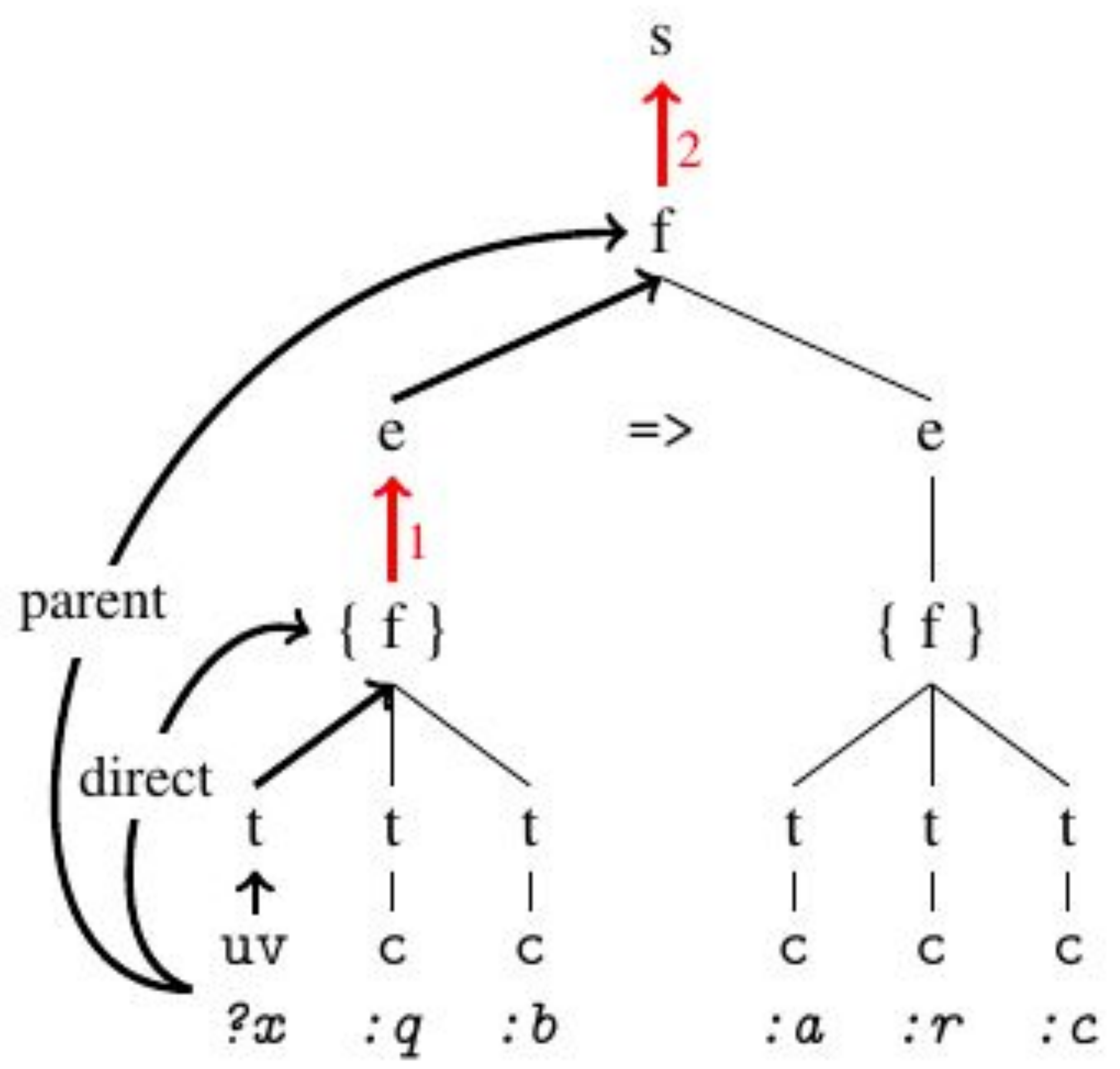


`s ::=`		start:
	`f`	formula
`f ::=`		formulas:
	`t t t.`	atomic formula
	`e => e.`	implication
	`f f`	conjunction
`t ::=`		terms:
	`uv`	universal variables
	`ex`	existential variables
	`c`	constants
	`e`	expressions
	`(k)`	lists
	`()`	empty list
`k ::=`		list content:
	`t`	term
	`t k`	term tail
`e ::=`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

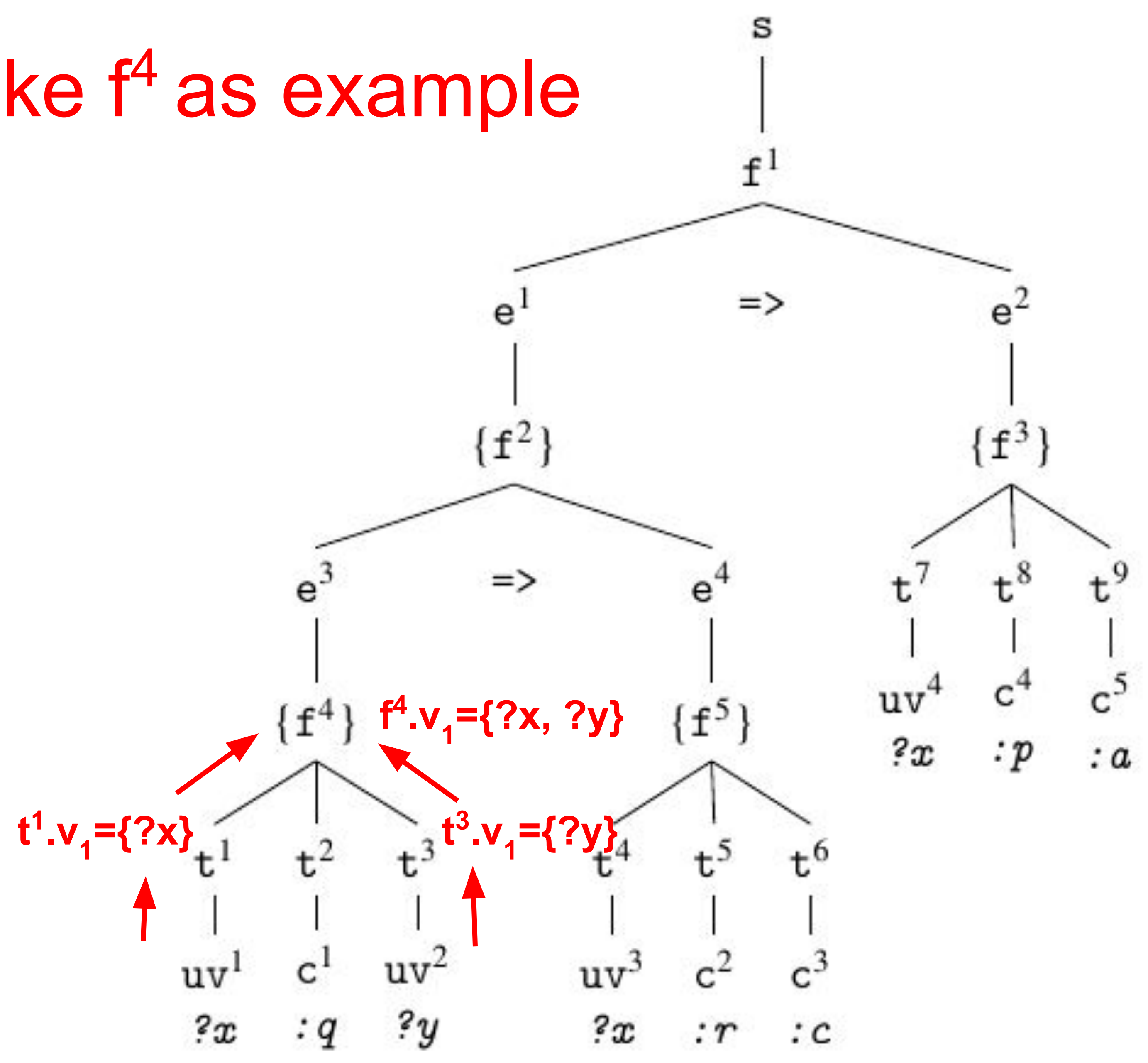
- attribute grammars provide a mechanism to pass information through a syntax tree of a context-free grammar
- to do so, **attributes** are defined on top of the production rules of the CFG
- attributes can either be **inherited** or **synthesized**
- **inherited attributes** pass information **downwards** 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 ::=	$t_1 t_2 t_3$.	$\boxed{f.v_1 \leftarrow t_1.v_1 \cup t_2.v_1 \cup t_3.v_1}$
	e_1=>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	$\boxed{t.v_1 \leftarrow \{uv\}}$
	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}=>{?x :r :c}}=>{?x :p :a}.

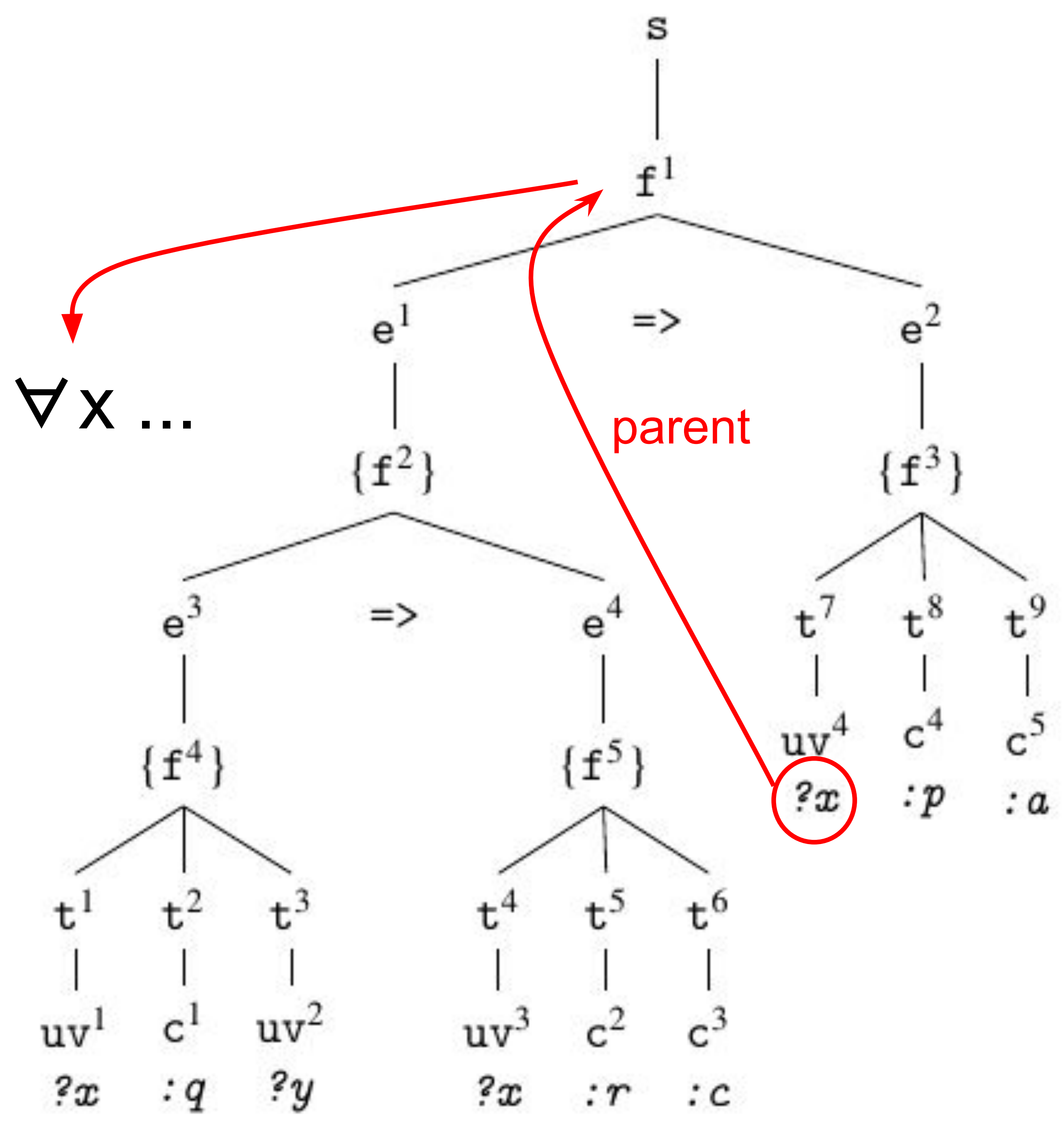
translates to

∀x. (∀y. <x q y>→<x r c>)→<x p a>

and **not** to

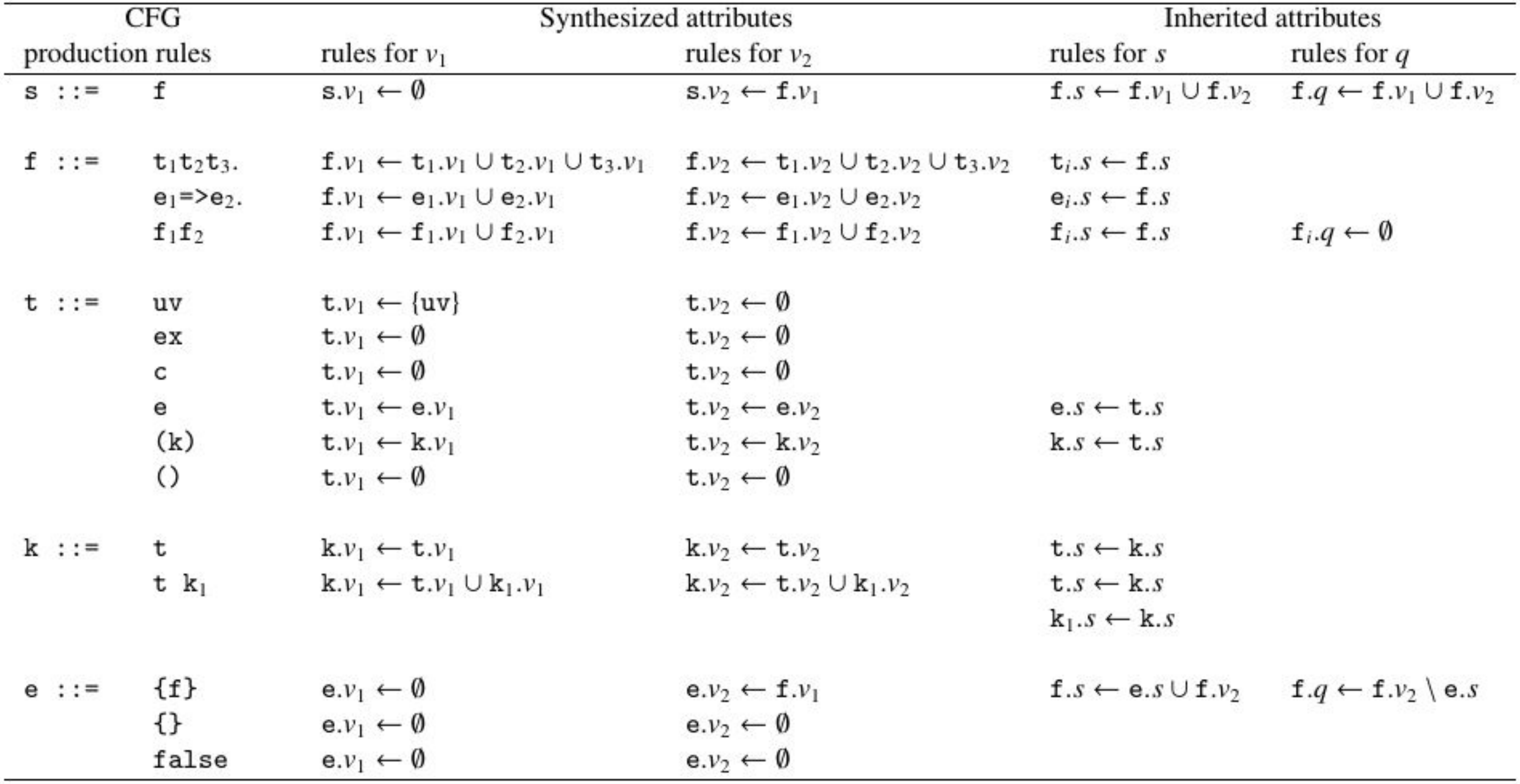
∀x. (∀x. ∀y. <x q y>→<x r c>)→<x p a>

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



GHENT UNIVERSITY imec

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`	$\mathtt{s}.v_1 \leftarrow \emptyset$	$\mathtt{s}.v_2 \leftarrow \mathtt{f}.v_1$	$\mathtt{f}.s \leftarrow \mathtt{f}.v_1 \cup \mathtt{f}.v_2$	$\mathtt{f}.q \leftarrow \mathtt{f}.v_1 \cup \mathtt{f}.v_2$
`f ::=`	$\mathtt{t_1 t_2 t_3.}$	$\mathtt{f}.v_1 \leftarrow \mathtt{t}_1.v_1 \cup \mathtt{t}_2.v_1 \cup \mathtt{t}_3.v_1$	$\mathtt{f}.v_2 \leftarrow \mathtt{t}_1.v_2 \cup \mathtt{t}_2.v_2 \cup \mathtt{t}_3.v_2$	$\mathtt{t}_i.s \leftarrow \mathtt{f}.s$	
	$\mathtt{e_1\text{=>}e_2.}$	$\mathtt{f}.v_1 \leftarrow \mathtt{e}_1.v_1 \cup \mathtt{e}_2.v_1$	$\mathtt{f}.v_2 \leftarrow \mathtt{e}_1.v_2 \cup \mathtt{e}_2.v_2$	$\mathtt{e}_i.s \leftarrow \mathtt{f}.s$	
	$\mathtt{f_1 f_2}$	$\mathtt{f}.v_1 \leftarrow \mathtt{f}_1.v_1 \cup \mathtt{f}_2.v_1$	$\mathtt{f}.v_2 \leftarrow \mathtt{f}_1.v_2 \cup \mathtt{f}_2.v_2$	$\mathtt{f}_i.s \leftarrow \mathtt{f}.s$	$\mathtt{f}_i.q \leftarrow \emptyset$
`t ::=`	`uv`	$\mathtt{t}.v_1 \leftarrow \{\mathtt{uv}\}$	$\mathtt{t}.v_2 \leftarrow \emptyset$		
	`ex`	$\mathtt{t}.v_1 \leftarrow \emptyset$	$\mathtt{t}.v_2 \leftarrow \emptyset$		
	`c`	$\mathtt{t}.v_1 \leftarrow \emptyset$	$\mathtt{t}.v_2 \leftarrow \emptyset$		
	`e`	$\mathtt{t}.v_1 \leftarrow \mathtt{e}.v_1$	$\mathtt{t}.v_2 \leftarrow \mathtt{e}.v_2$	$\mathtt{e}.s \leftarrow \mathtt{t}.s$	
	`(k)`	$\mathtt{t}.v_1 \leftarrow \mathtt{k}.v_1$	$\mathtt{t}.v_2 \leftarrow \mathtt{k}.v_2$	$\mathtt{k}.s \leftarrow \mathtt{t}.s$	
	`()`	$\mathtt{t}.v_1 \leftarrow \emptyset$	$\mathtt{t}.v_2 \leftarrow \emptyset$		
`k ::=`	`t`	$\mathtt{k}.v_1 \leftarrow \mathtt{t}.v_1$	$\mathtt{k}.v_2 \leftarrow \mathtt{t}.v_2$	$\mathtt{t}.s \leftarrow \mathtt{k}.s$	
	$\mathtt{t\ k_1}$	$\mathtt{k}.v_1 \leftarrow \mathtt{t}.v_1 \cup \mathtt{k}_1.v_1$	$\mathtt{k}.v_2 \leftarrow \mathtt{t}.v_2 \cup \mathtt{k}_1.v_2$	$\mathtt{t}.s \leftarrow \mathtt{k}.s$	
				$\mathtt{k}_1.s \leftarrow \mathtt{k}.s$	
`e ::=`	`{f}`	$\mathtt{e}.v_1 \leftarrow \emptyset$	$\mathtt{e}.v_2 \leftarrow \mathtt{f}.v_1$	$\mathtt{f}.s \leftarrow \mathtt{e}.s \cup \mathtt{f}.v_2$	$\mathtt{f}.q \leftarrow \mathtt{f}.v_2 \setminus \mathtt{e}.s$
	`{}`	$\mathtt{e}.v_1 \leftarrow \emptyset$	$\mathtt{e}.v_2 \leftarrow \emptyset$		
	`false`	$\mathtt{e}.v_1 \leftarrow \emptyset$	$\mathtt{e}.v_2 \leftarrow \emptyset$		

Outline

Notation3 Logic

Implicit Quantification

N3 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

GHENT UNIVERSITY

imec

IDLab

INTERNET & DATA LAB