Datalog as a Uniform Framekwork for Parsing and Generation

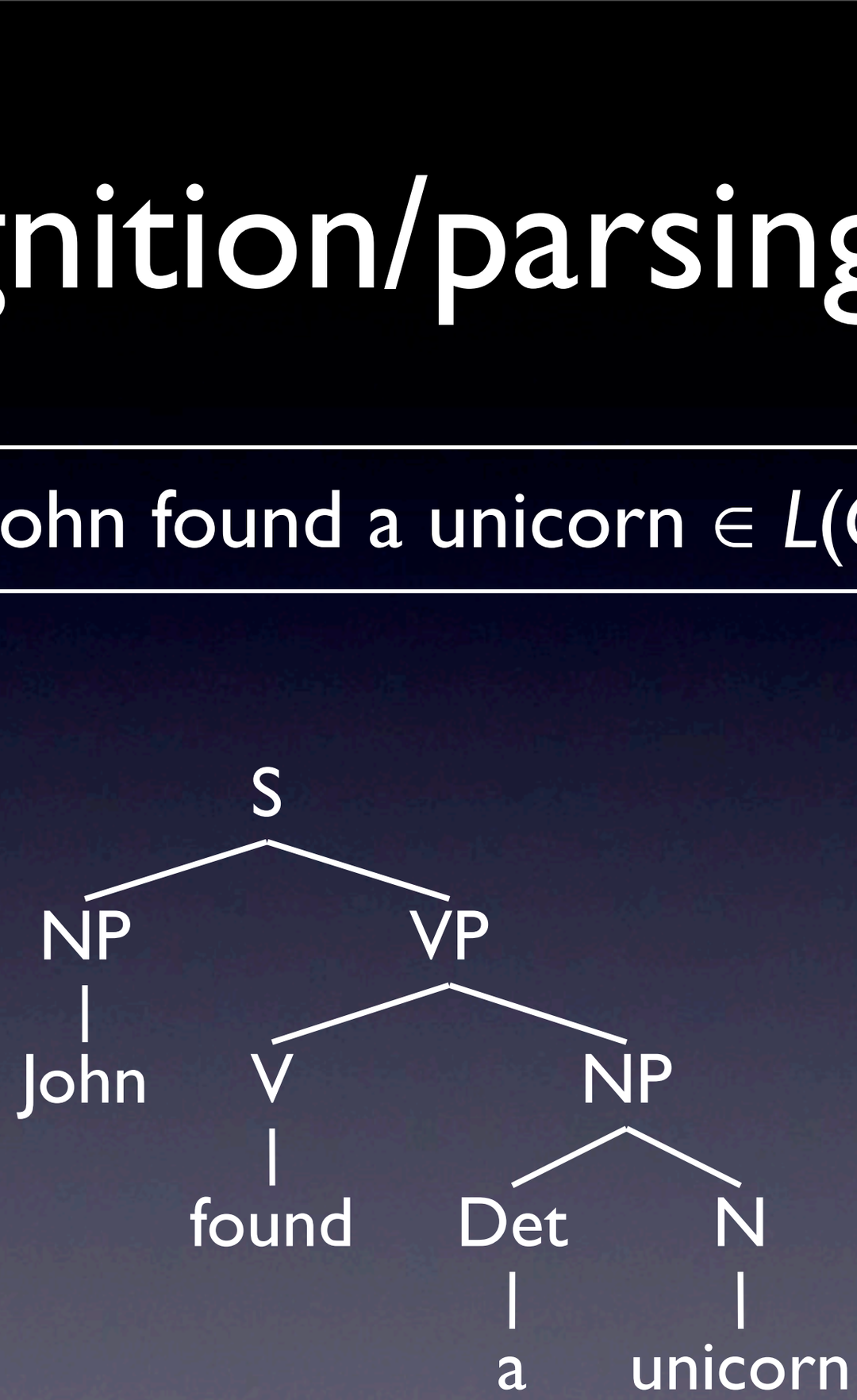
Makoto Kanazawa
National Institute of Informatics
Tokyo, Japan

Reduction of various “context-free grammar” formalisms to Datalog
Look at complexity-theoretic consequences

CFG recognition/parsing

S → NP VP
VP → V NP
NP → Det N
NP → John
V → found
Det → a
N → unicorn

John found a unicorn ∈ *L*(*G*) ?



To give an idea of what the reduction looks like
Well-known case of CFGs

Datalog query evaluation

```
S(i, k) :- NP(i, j), VP(j, k).
VP(i, k) :- V(i, j), NP(j, k).
NP(i, k) :- Det(i, j), N(j, k).
NP(i, j) :- John(i, j).
V(i, j) :- found(i, j).
Det(i, j) :- a(i, j).
N(i, j) :- unicorn(i, j).
```

program

```
John(0, 1).
found(1, 2).
a(2, 3).
unicorn(3, 4).
```

database

```
?- S(0, 4).
```

query

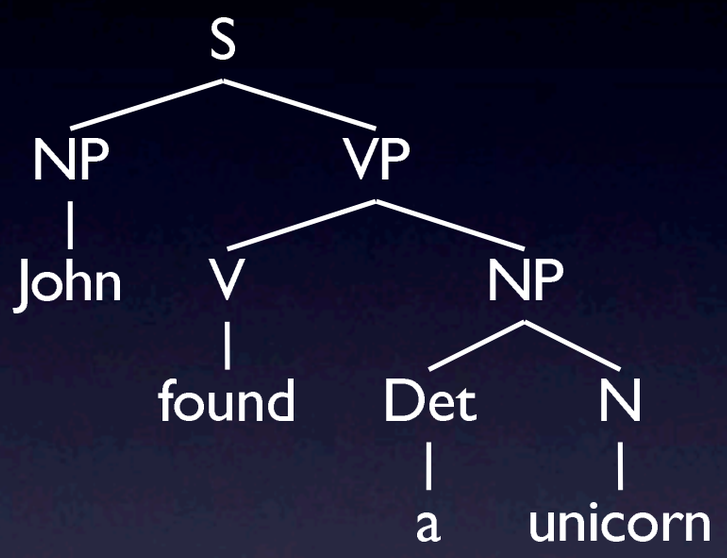
Definite clause grammar representation
Executable as Prolog code

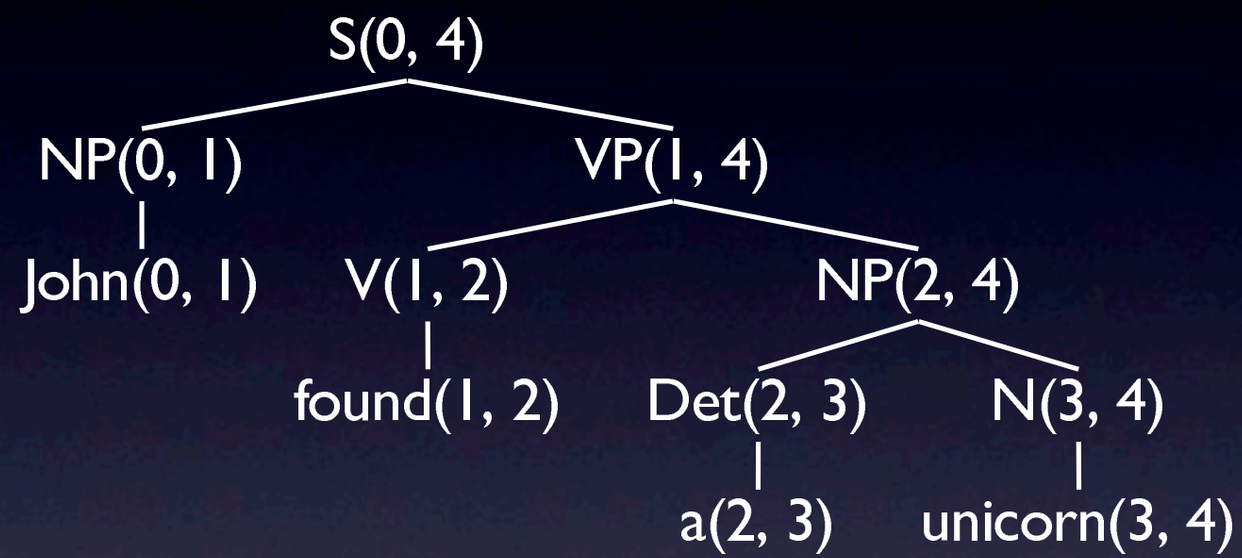
John(0, 1) found(1, 2) a(2, 3) unicorn(3, 4)

S(*i*, *k*) :- NP(*i*, *j*), VP(*j*, *k*).

?- S(0, 4).

The conversion is very straightforward
String -> string graph





CFG recognition/parsing ≈ Datalog query evaluation

CFG derivation tree and Datalog derivation tree isomorphic to each other
Finding one amounts to finding the other

Parsing and generation as Datalog query evaluation

Kanazawa 2007

Recognition/Parsing	CFG MCFG RTG MRTG $CFTG_{IO}$ ⋮
Generation	CFG + Montague semantics†

† with a certain restriction

Parsing and generation as Datalog query evaluation

- Algorithms
  - Seminaive bottom-up ≈ CYK
  - Magic-sets rewriting ≈ Earley
- Computational complexity
  - Fixed grammar recognition
  - Uniform recognition
  - Parsing

Polynomial-time algorithm

```
Seminaive(P, D)
1   agenda[0] ← D
2   i ← 0
3   chart ← ∅
4   while agenda[i] ≠ ∅
5         do
6               chart ← chart ∪ agenda[i]
7               agenda[i + 1] ← Conseq(P, agenda[i], chart) − chart
8               i ← i + 1
9   return chart
```

holds facts with derivation tree of minimal height *i*

facts that immediately follow from one fact in *agenda*[*i*] plus some facts in *chart*

well-formed substring table

Computational complexity

- CFG

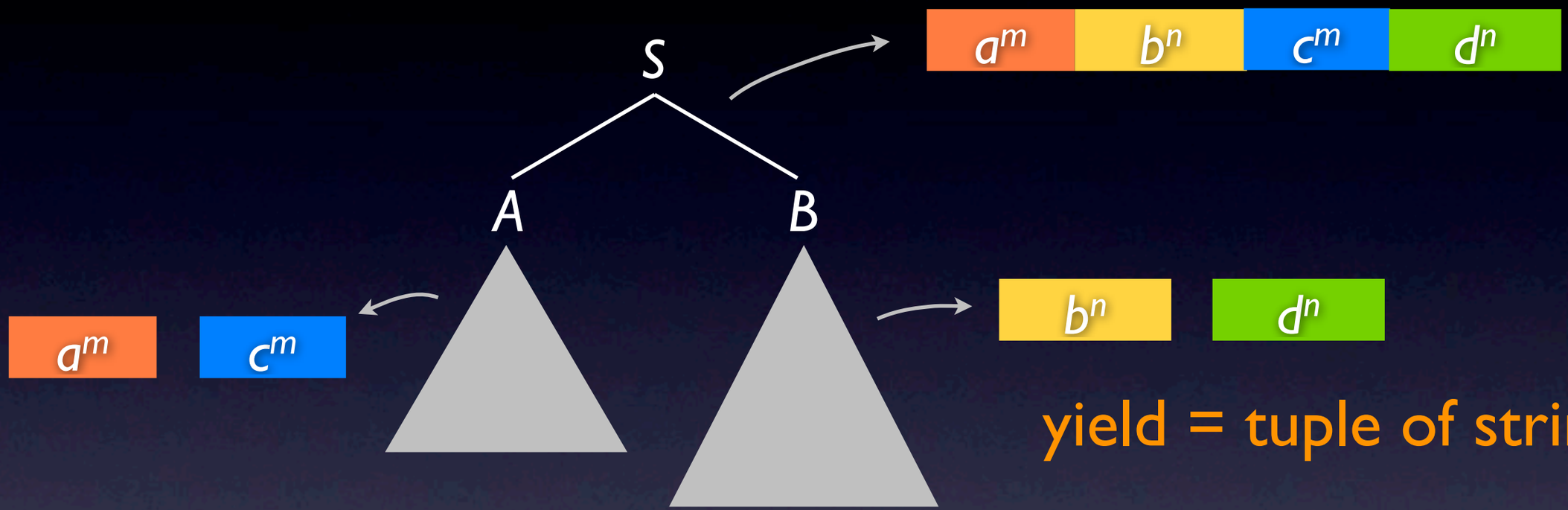
Fixed grammar recognition	LOGCFL-complete
Uniform recognition	P-complete
ε-free uniform recognition	LOGCFL-complete

LOGCFL

- The class of problems that reduce to some context-free language in logarithmic space
- The smallest computational complexity class that includes the context-free languages

$$AC^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq \text{LOGCFL} \subseteq AC^1 \subseteq NC^2 \subseteq P \subseteq NP$$

Multiple context-free grammars



derivation tree

$$S(x_1 y_1 x_2 y_2) :- A(x_1, x_2), B(y_1, y_2).$$

$$A(\varepsilon, \varepsilon). \qquad B(\varepsilon, \varepsilon).$$

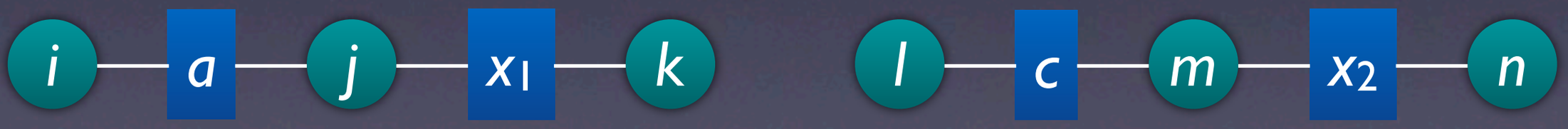
$$A(a x_1, c x_2) :- A(x_1, x_2). \qquad B(b y_1, d y_2) :- B(y_1, y_2).$$

This is an example of a 2-MCFG.
An m-MCFG allows nonterminals to take up to m arguments.

$$S(x_1 y_1 x_2 y_2) :- A(x_1, x_2), B(y_1, y_2).$$

$$S(i, m) :- A(i, j, k, l), B(j, k, l, m).$$

$$A(ax_1, cx_2) :- A(x_1, x_2).$$



$$A(i, k, l, n) :- A(j, k, m, n), a(i, j), a(l, m).$$

Range concatenation grammar

- Syntax of EFS (*aka* LMG)
- Variables interpreted as ranges (pairs of string positions)
- Semantics ≈ resolution proof of Datalog
- Can be regarded as a subset of Datalog
- Captures P on strings (just like Datalog)
- I focus on a LOGCFL subset of Datalog (on strings, trees, and λ-terms)

CFG + Montague semantics

$S(X_1 X_2) \rightarrow NP(X_1)\ VP(X_2)$

$VP(\lambda x.X_2(\lambda y.X_1 y x)) \rightarrow V(X_1)\ NP(X_2)$

$V(\lambda y x.X_2(X_1 y x)(X_3 y x)) \rightarrow V(X_1)\ Conj(X_2)\ V(X_3)$

$NP(X_1 X_2) \rightarrow Det(X_1)\ N(X_2)$

$NP(\lambda u.u\ \mathbf{John}^{e}) \rightarrow$ John

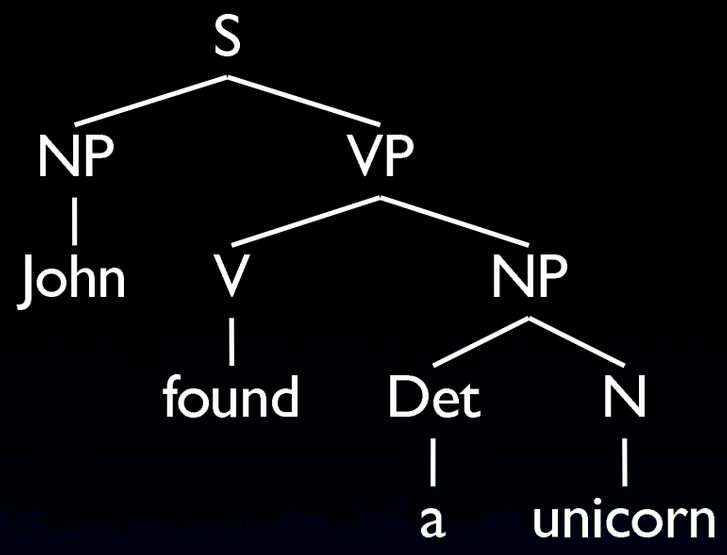
$V(\mathbf{find}^{e\rightarrow e\rightarrow t}) \rightarrow$ found

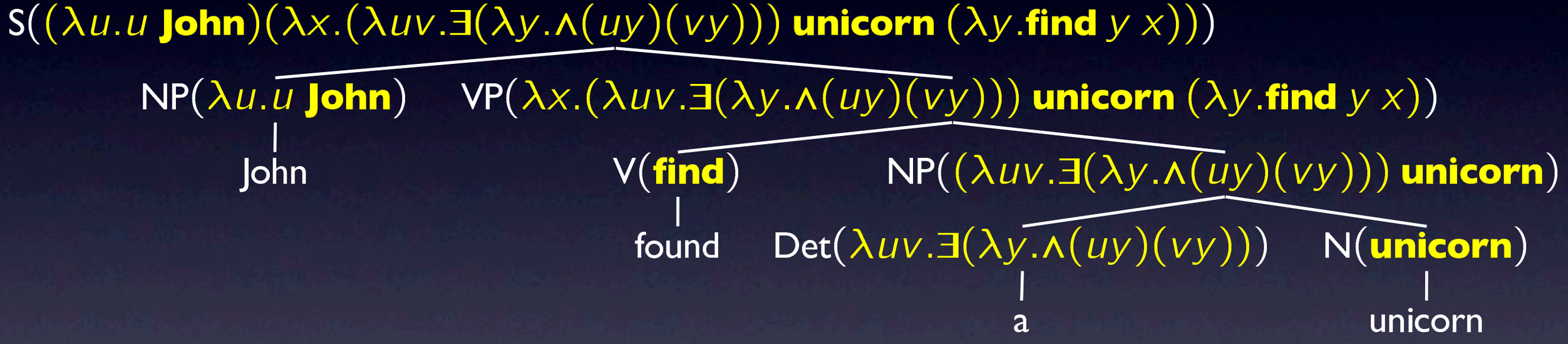
$V(\mathbf{catch}^{e\rightarrow e\rightarrow t}) \rightarrow$ caught

$Conj(\wedge^{t\rightarrow t\rightarrow t}) \rightarrow$ and $\qquad \lambda uv.\exists y(u(y) \wedge v(y))$

$Det(\lambda uv.\exists^{(e\rightarrow t)\rightarrow t}(\lambda y.\wedge^{t\rightarrow t\rightarrow t}(uy)(vy))) \rightarrow$ a

$N(\mathbf{unicorn}^{e\rightarrow t}) \rightarrow$ unicorn





$$(\lambda u.u\ \textbf{John})(\lambda x.(\lambda uv.\exists(\lambda y.\wedge(uy)(vy))\ \textbf{unicorn}\ (\lambda y.\textbf{find}\ y\ x)))$$

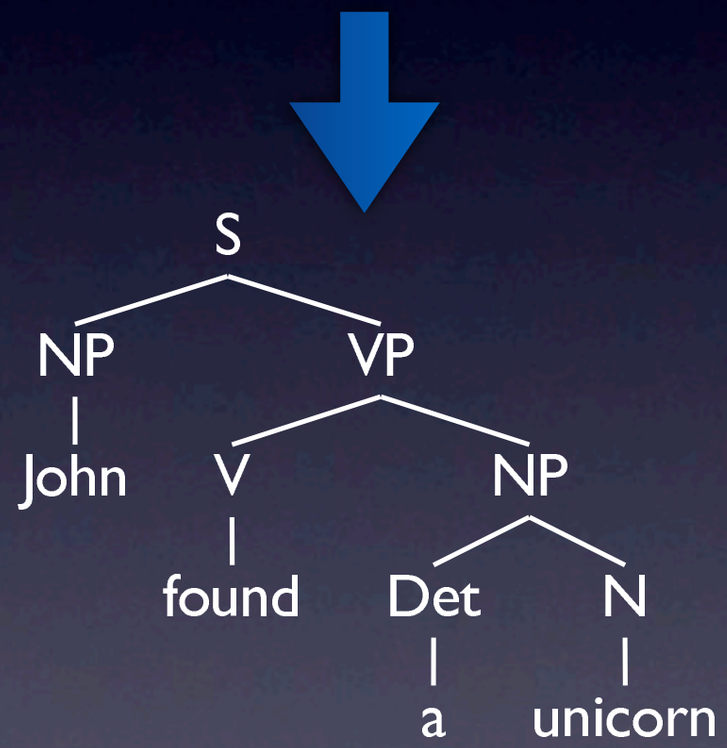
$$\twoheadrightarrow_\beta\ \exists(\lambda y.\wedge(\textbf{unicorn}\ y)(\textbf{find}\ y\ \textbf{John}))$$

$$\approx\ \exists y(\textbf{unicorn}(y) \wedge \textbf{find}(\textbf{John}, y))$$

logical form

Surface realization

$\exists y(\textbf{unicorn}(y) \wedge \textbf{find}(\textbf{John}, y))$ input logical form



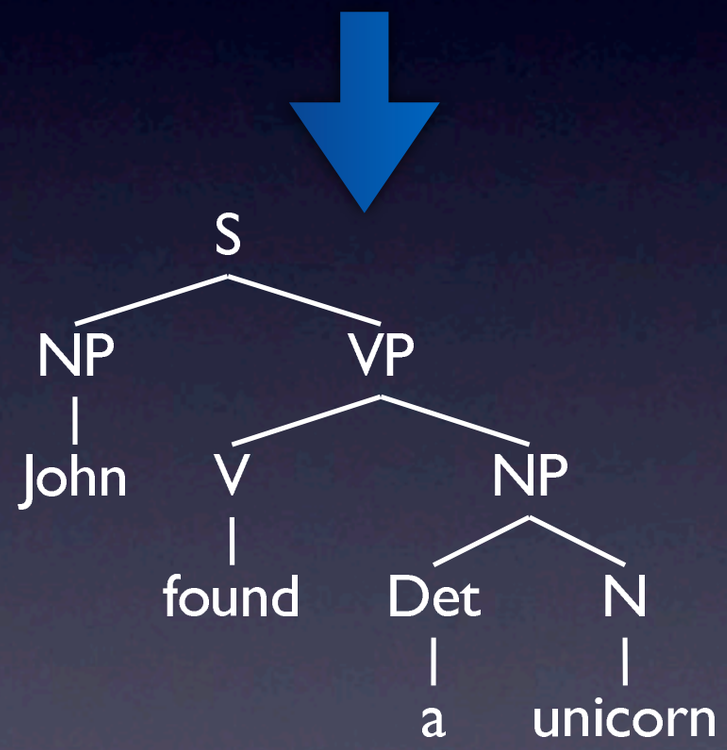
derivation tree

John found a unicorn

output surface form

Parsing of input logical form

$\exists y(\textbf{unicorn}(y) \wedge \textbf{find}(\textbf{John}, y))$ input logical form



derivation tree

Recognition of surface realizability

$\exists y(\textbf{unicorn}(y) \wedge \textbf{find}(\textbf{John}, y))$ input logical form

surface realizable?

yes/no

Context-free grammars on λ-terms

$\mathrm{S}(X_1 X_2) \coloneq \mathrm{NP}(X_1)\ \mathrm{VP}(X_2)$

$\mathrm{VP}(\lambda x.X_2(\lambda y.X_1 y x)) \coloneq \mathrm{V}(X_1), \mathrm{NP}(X_2).$

$\mathrm{V}(\lambda y x.X_2(X_1 y x)(X_3 y x)) \coloneq \mathrm{V}(X_1), \mathrm{Conj}(X_2), \mathrm{V}(X_3).$

$\mathrm{NP}(X_1 X_2) \coloneq \mathrm{Det}(X_1), \mathrm{N}(X_2).$

$\mathrm{NP}(\lambda u.u\ \mathbf{John}^{e}).$

$\mathrm{V}(\mathbf{find}^{e\to e\to t}).$

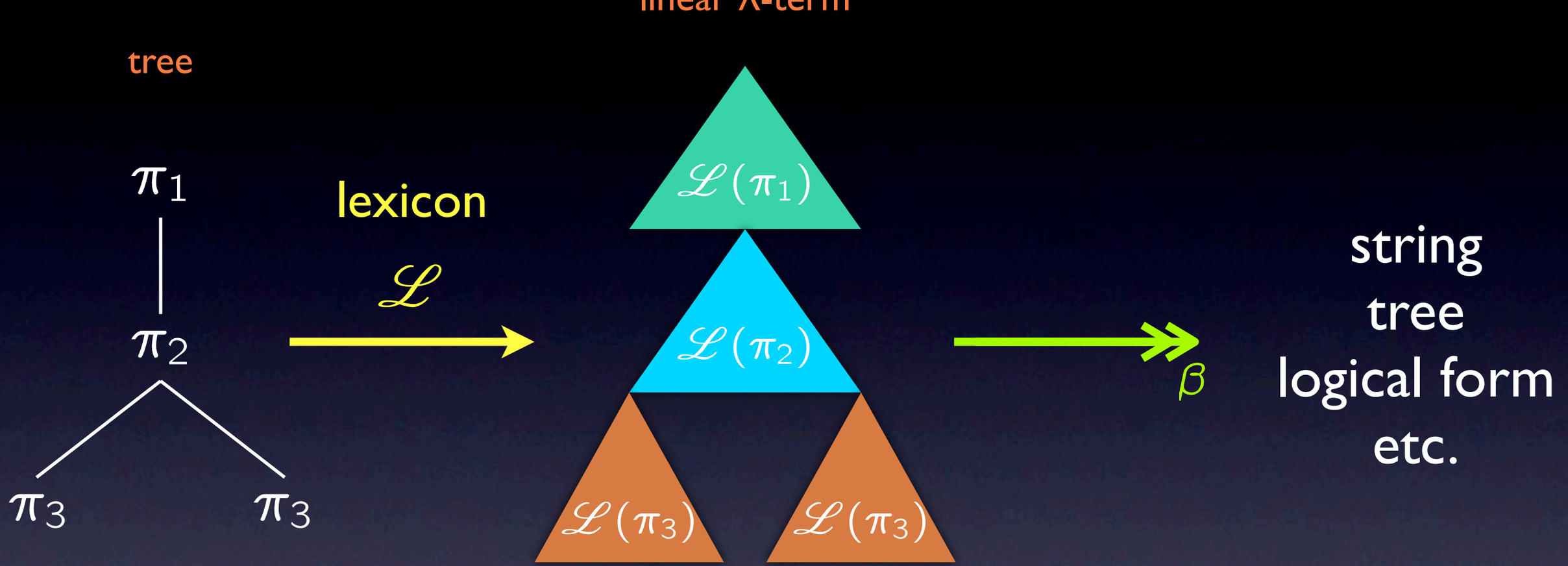
$\mathrm{V}(\mathbf{catch}^{e\to e\to t}).$

$\mathrm{Conj}(\wedge^{t\to t\to t}).$

$\mathrm{Det}(\lambda uv.\exists^{(e\to t)\to t}(\lambda y.\wedge^{t\to t\to t}(uy)(vy))).$

$\mathrm{N}(\mathbf{unicorn}^{e\to t}).$

Second-order ACGs



Context-free grammars on λ-terms = second-order non-linear ACGs

$$\pi\colon B(M) \coloneq B_1(X_1),\ldots,B_n(X_n).$$

$$\pi : B_1 \to \cdots \to B_n \to B : \lambda X_1 \ldots X_n . M$$

abstract constant

type of π

object realization of π

From CFLG recognition to Datalog query evaluation

$S(X_1 X_2) :\!- NP(X_1)\ VP(X_2)$

$VP(\lambda x.X_2(\lambda y.X_1 y x)) :\!- V(X_1), NP(X_2).$

$V(\lambda y x.X_2(X_1 y x)(X_3 y x)) :\!- V(X_1), Conj(X_2), V(X_3).$

$NP(X_1 X_2) :\!- Det(X_1), N(X_2).$

$NP(\lambda u.u\ \mathbf{John}^{e}).$

$V(\mathbf{find}^{e\to e\to t}).$

$V(\mathbf{catch}^{e\to e\to t}).$

$Conj(\wedge^{t\to t\to t}).$

$Det(\lambda uv.\exists^{(e\to t)\to t}(\lambda y.\wedge^{t\to t\to t}(uy)(vy))).$

$N(\mathbf{unicorn}^{e\to t}).$

$\exists(\lambda y.\wedge(\mathbf{unicorn}\ y)(\mathbf{find}\ y\ \mathbf{John})) \in L(G)?$

From CFLG recognition to Datalog query evaluation

$S(i_1) \coloneq NP(i_1, i_2, i_3)\ VP(i_2, i_3)$

$VP(i_1, i_4) \coloneq V(i_2, i_4, i_3), NP(i_1, i_2, i_3).$

$V(i_1, i_4, i_3) \coloneq V(i_2, i_4, i_3), Conj(i_1, i_5, i_2), V(i_5, i_4, i_3).$

$NP(i_1, i_4, i_5) \coloneq Det(i_1, i_4, i_5, i_2, i_3), N(i_2, i_3).$

$NP(i_1, i_1, i_2) \coloneq \mathbf{John}(i_2).$

$V(i_1, i_3, i_2) \coloneq \mathbf{find}(i_1, i_3, i_2).$

$V(i_1, i_3, i_2) \coloneq \mathbf{catch}(i_1, i_3, i_2).$

$Conj(i_1, i_3, i_2) \coloneq \wedge(i_1, i_3, i_2).$

$Det(i_1, i_5, i_4, i_3, i_4) \coloneq \exists(i_1, i_2, i_4), \wedge(i_2, i_5, i_3).$

$N(i_1, i_2) \coloneq \mathbf{unicorn}(i_1, i_2).$

$\exists(1, 2, 4).$

$\wedge(2, 5, 3).$

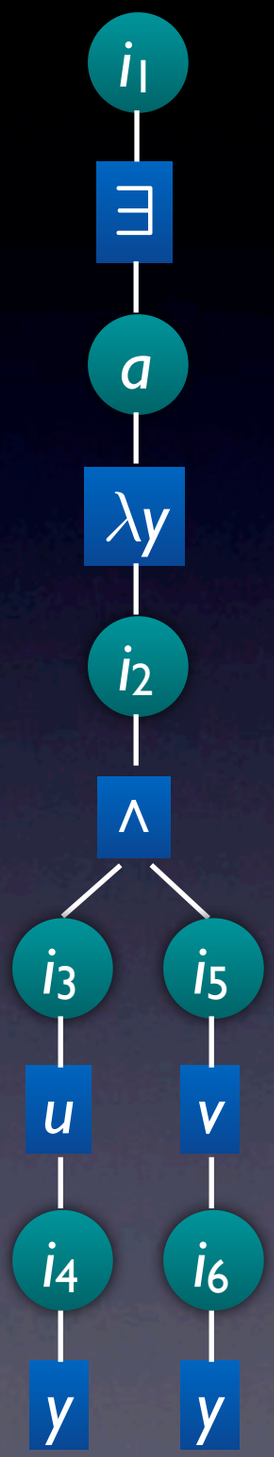
$\mathbf{unicorn}(3, 4).$

$\mathbf{find}(5, 6, 4).$

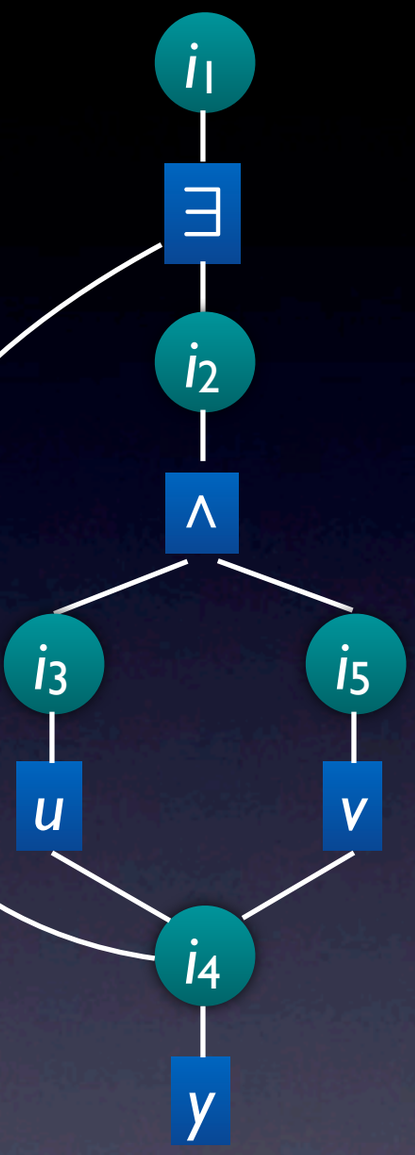
$\mathbf{John}(6).$

$?\!- S(1).$

$\exists(\lambda y.\wedge(uy)(vy))$

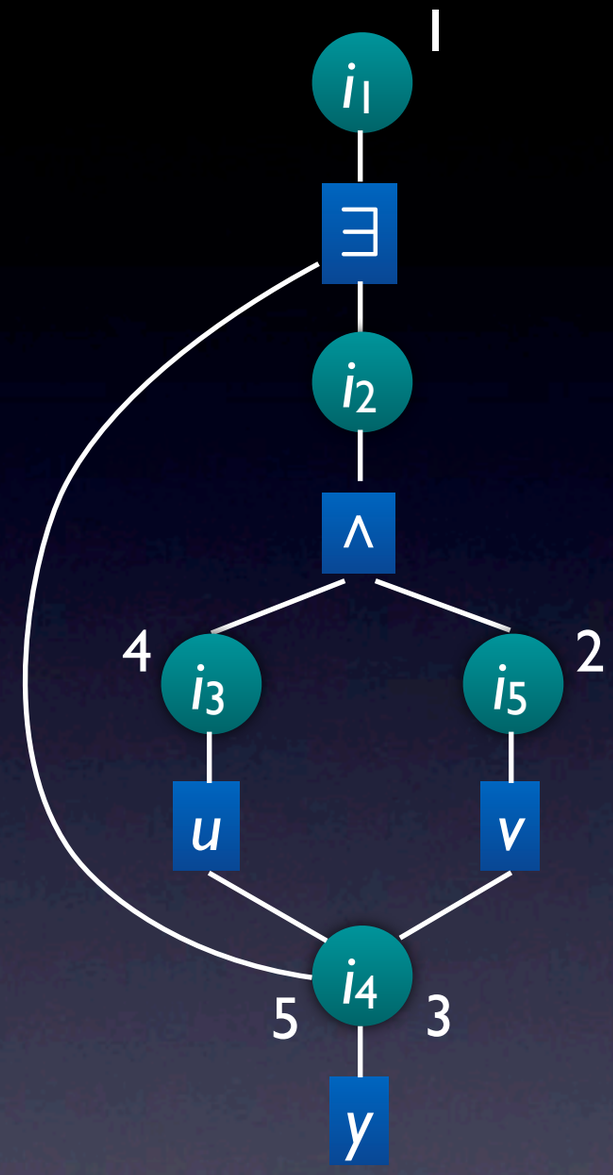


tree graph



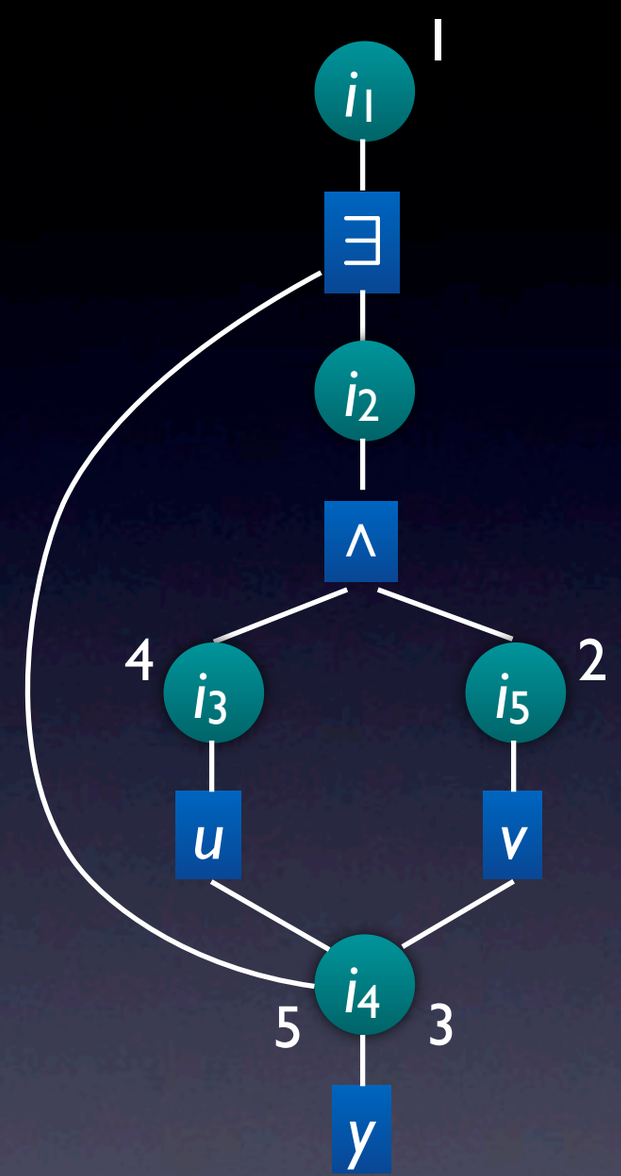
term graph

$\lambda uv.\exists(\lambda y.\wedge(uy)(vy))$



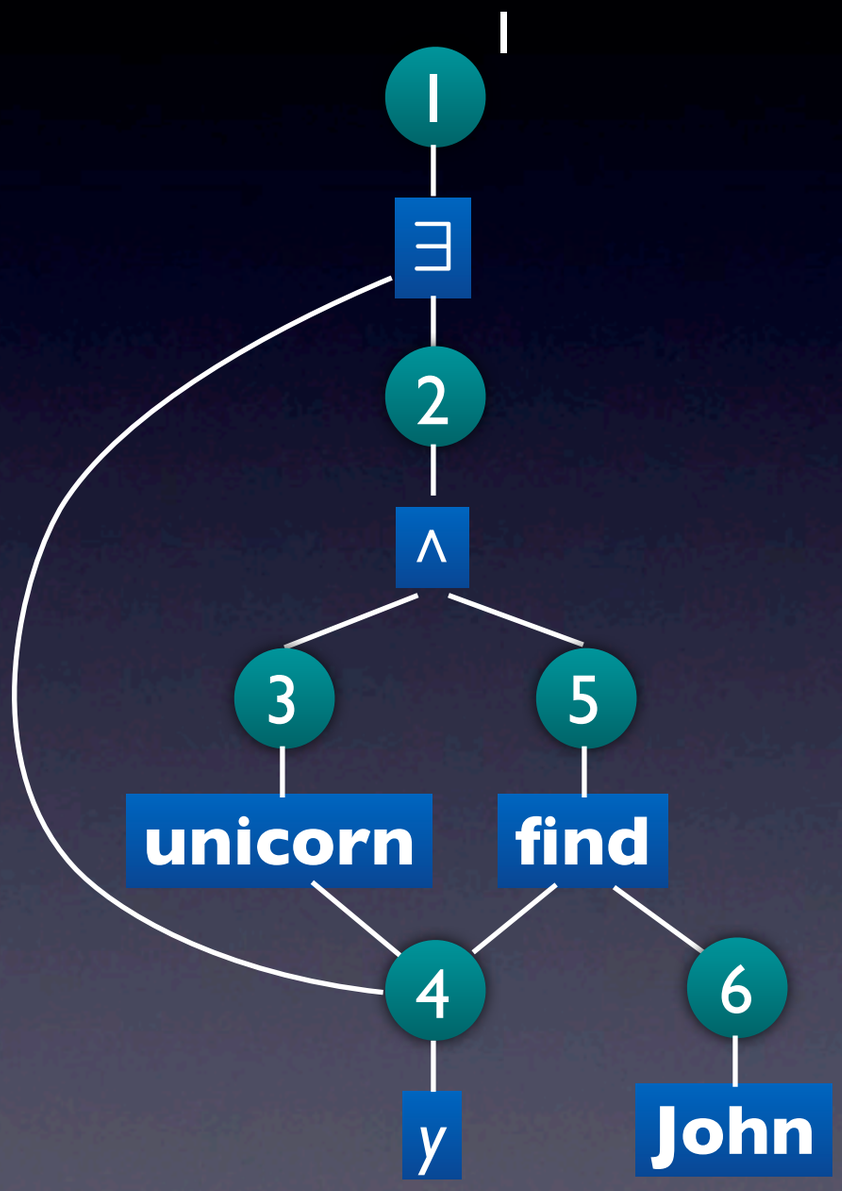
term graph with external nodes

$$\mathrm{Det}(\lambda uv.\exists^{(e\to t)\to t}(\lambda y.\wedge^{t\to t\to t}(uy)(vy))).$$



$$\mathrm{Det}(i_1, i_5, i_4, i_3, i_4) :- \exists(i_1, i_2, i_4), \wedge(i_2, i_5, i_3).$$

$\exists(\lambda y.\wedge(\textbf{unicorn}\, y)(\textbf{find}\, y\, \textbf{John}))$




```
∃(1, 2, 4).
∧(2, 5, 3).
unicorn(3, 4).
find(5, 6, 4).
John(6).
```

```
?– S(1).
```

From CFLG recognition to Datalog query evaluation

- The reduction is correct when all λ-terms in the grammar are almost linear.

$$\frac{}{x : \alpha \vdash x : \alpha} \qquad \frac{}{\vdash c : \tau(c)}$$

$$\frac{\Gamma \vdash M : \alpha \quad \Delta \vdash N : \beta}{\Gamma \cup \Delta \vdash MN : \beta} \text{ if } \Gamma \cap \Delta \subseteq \mathit{At} \qquad \frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \to \beta}$$

Almost linear λ-terms

- Coherence Theorem: An almost linear λ-term is determined by its principal typing up to βη-equality.
- Subject Expansion Theorem: An almost linear λ-term has the same principal typing as its β-normal form.

Parsing and generation as Datalog query evaluation

Kanazawa 2007

Recognition/ Parsing	CFG MCFG RTG MRTG $CFTG_{IO}$ ⋮
Generation	CFG, etc. + Montague semantics†

→ Recognition/ Parsing: almost linear CFLGs → Datalog query evaluation

† when almost linear

Computational complexity

- Almost linear CFLGs with bounded width and rank

Fixed grammar recognition	LOGCFL-complete
Uniform recognition	P-complete
ε-free uniform recognition	LOGCFL-complete

width = arity of Datalog predicates
rank = number of subgoals in rules
ε-rule = Datalog rule with empty right-hand side

LOGCFL

- Characterized in terms of alternating Turing machines operating in log space with polynomial-size accepting computation trees

Ruzzo 1980

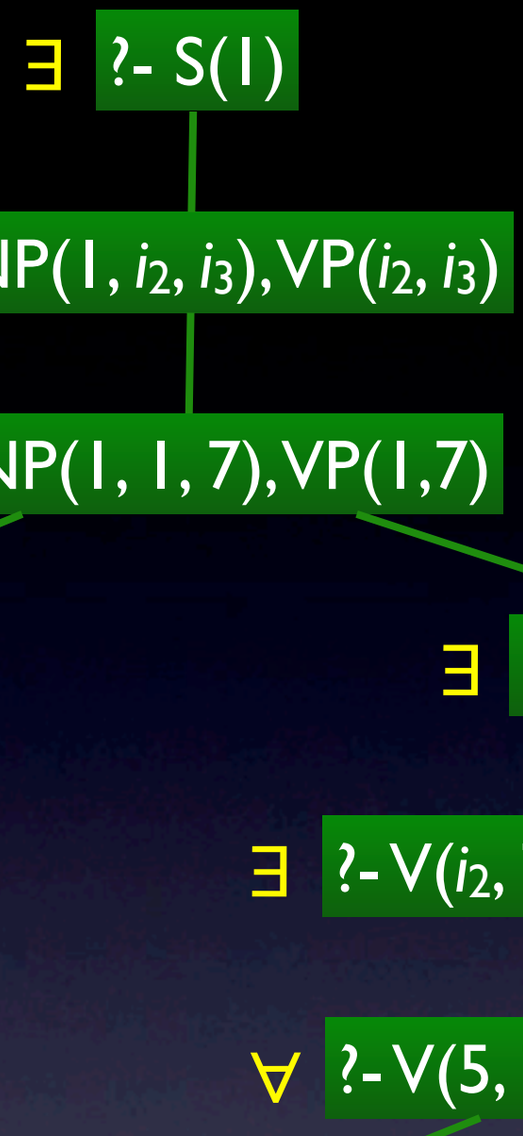
Alternating algorithm for Datalog query evaluation

- Each configuration is an attempt to prove some ground fact

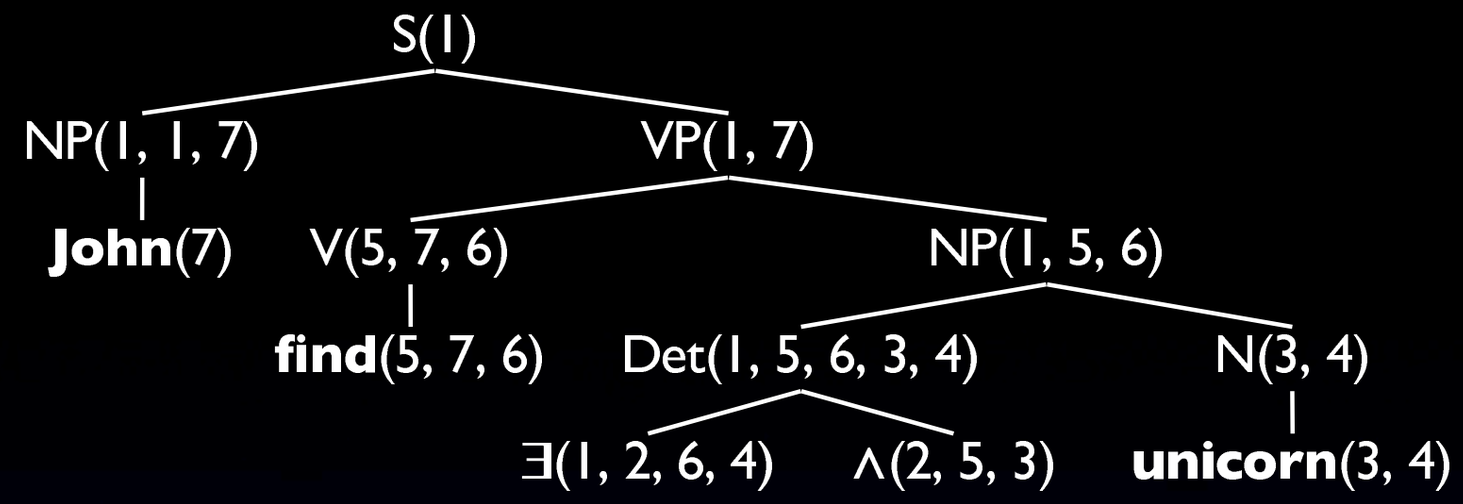
∃
- Pick a matching rule

∃
- Pick constants that instantiate the rule

∀
- Prove all subgoals of the instantiated rule
- Can operate in log space if width and rank are bounded



accepting computation tree



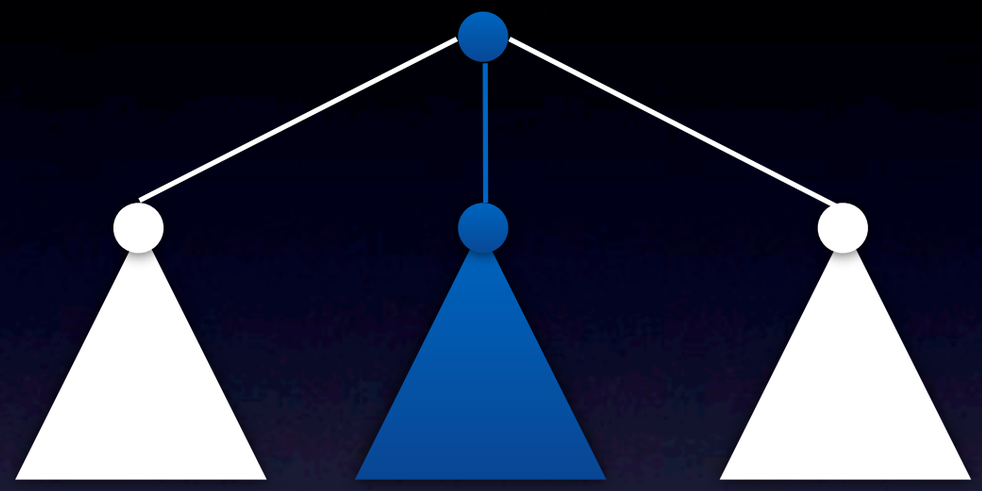
Datalog derivation tree

Size of Datalog derivation tree

Type 0

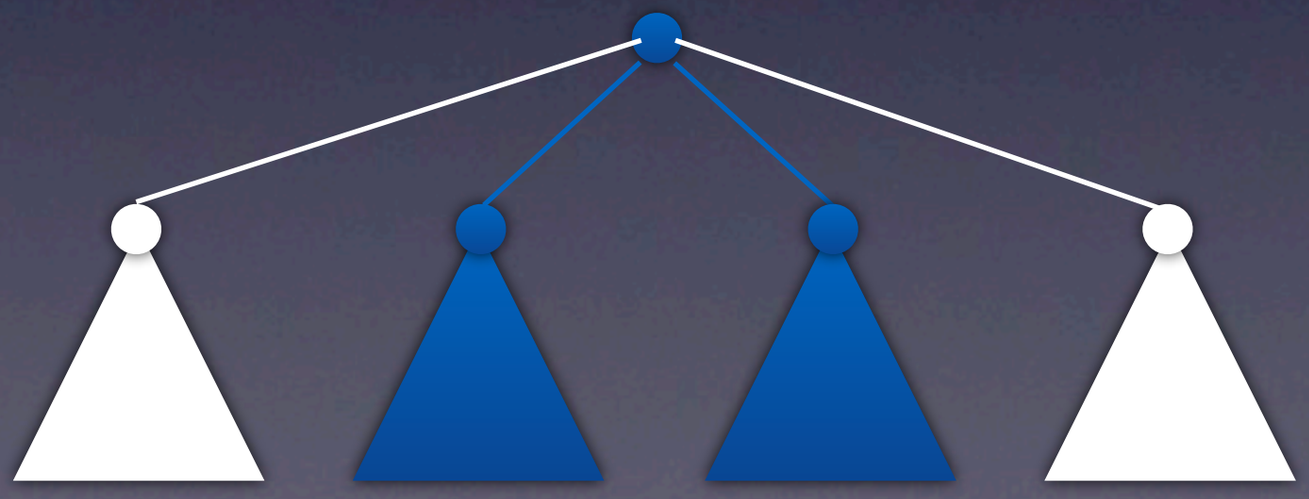
no database facts below

Type 1



all children except one are Type 0

Type 2



at least two children that are not Type 0

Number of nodes

n = number of extensional nodes
(nodes labeled by database facts)

- Type 0: exponential in the number of predicates, polynomial in *n*
- Type 1: linear in the number of predicates, polynomial in *n*
- Type 2: at most *n*−1

Alternating algorithm for almost linear CFLG recognition

G *M* β-normal form

log space

program(*G*) (database(*M*), query(*M*))

query(*M*)

derivation tree
from program(*G*) ∪ query(*M*)

The number of extensional nodes bounded by the size of *M*

Complexity of parsing

- Almost linear CFLGs with bounded width and rank
- Output: parse forest = set of all ground rule instances useful for deriving the input query

Fixed grammar	FL^{LOGCFL}
Uniform	FP
ε-free uniform	FL^{LOGCFL}