

Symmetric Datalog \neq Linear Datalog

László Egri¹,

joint work with Benoît Larose² and Pascal Tesson³

¹McGill University

²Concordia University

³Université Laval

UA and CSP, Nashville June 2007

Introduction

- Symmetric Datalog (LE, Larose, Tesson, 2007)

Introduction

- Symmetric Datalog (LE, Larose, Tesson, 2007)
 - ◆ In logarithmic space (using Reingold, 2005)

Introduction

- Symmetric Datalog (LE, Larose, Tesson, 2007)
 - ◆ In logarithmic space (using Reingold, 2005)
 - ◆ Boolean domains + standard complexity assumptions → all CSPs in L are in symmetric Datalog

Introduction

- Symmetric Datalog (LE, Larose, Tesson, 2007)
 - ◆ In logarithmic space (using Reingold, 2005)
 - ◆ Boolean domains + standard complexity assumptions → all CSPs in L are in symmetric Datalog
 - ◆ Conjecture: all CSPs in L are in symmetric Datalog

Introduction

- Symmetric Datalog (LE, Larose, Tesson, 2007)
 - ◆ In logarithmic space (using Reingold, 2005)
 - ◆ Boolean domains + standard complexity assumptions → all CSPs in L are in symmetric Datalog
 - ◆ Conjecture: all CSPs in L are in symmetric Datalog
- Undirected *st*-connectivity is definable in symmetric Datalog

Introduction

- Symmetric Datalog (LE, Larose, Tesson, 2007)
 - ◆ In logarithmic space (using Reingold, 2005)
 - ◆ Boolean domains + standard complexity assumptions → all CSPs in L are in symmetric Datalog
 - ◆ Conjecture: all CSPs in L are in symmetric Datalog
- Undirected st -connectivity is definable in symmetric Datalog
- Directed st -connectivity is not definable in symmetric Datalog

Introduction

- Symmetric Datalog (LE, Larose, Tesson, 2007)
 - ◆ In logarithmic space (using Reingold, 2005)
 - ◆ Boolean domains + standard complexity assumptions → all CSPs in L are in symmetric Datalog
 - ◆ Conjecture: all CSPs in L are in symmetric Datalog
- Undirected st -connectivity is definable in symmetric Datalog
- Directed st -connectivity is not definable in symmetric Datalog
 - ◆ Reflexive transitive closure of a binary relation is not definable in symmetric Datalog

Introduction

- Symmetric Datalog (LE, Larose, Tesson, 2007)
 - ◆ In logarithmic space (using Reingold, 2005)
 - ◆ Boolean domains + standard complexity assumptions \rightarrow all CSPs in L are in symmetric Datalog
 - ◆ Conjecture: all CSPs in L are in symmetric Datalog
- Undirected *st*-connectivity is definable in symmetric Datalog
- Directed *st*-connectivity is not definable in symmetric Datalog
 - ◆ Reflexive transitive closure of a binary relation is not definable in symmetric Datalog
 - ◆ $\neg\text{CSP}(\langle\{0, 1\}; \leq, \{0\}, \{1\}\rangle)$ is not definable in symmetric Datalog

- Recap symmetric Datalog through an example

Outline

- Recap symmetric Datalog through an example
- Definitions: **free derivation path, the free structure**

Outline

- Recap symmetric Datalog through an example
- Definitions: **free derivation path, the free structure**
- Overview of the general proof

Outline

- Recap symmetric Datalog through an example
- Definitions: **free derivation path, the free structure**
- Overview of the general proof
- The main idea through an example

Datalog and Derivation Path Example

Input Vocabulary:

S^1, T^1, E^2

Linear (Symmetric) Program:

EDB: Extensional Database Predicate

IDB: Intensional Database Predicate

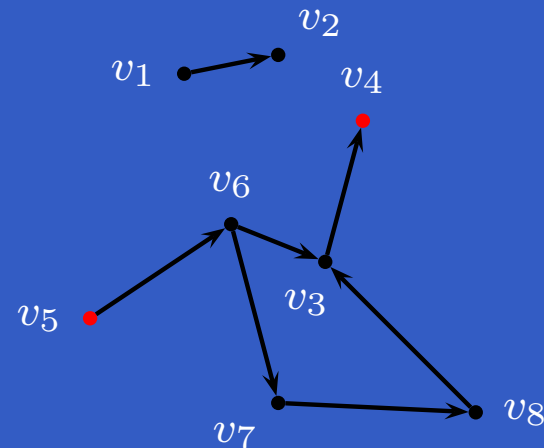
$I(y) \leftarrow S(y)$

$I(y) \leftarrow I(x); E(x, y)$

$(I(x) \leftarrow I(y); E(x, y))$

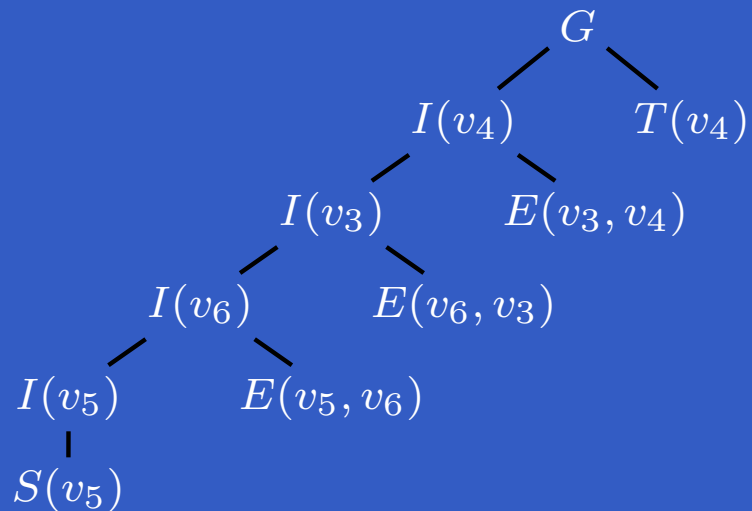
$G \leftarrow I(y); T(y)$

Input Structure:



$S = \{v_5\}, T = \{v_4\}$

Derivation Path:



The Free Derivation Path

Symmetric Program \mathcal{D} :

$$I(y) \leftarrow S(y)$$

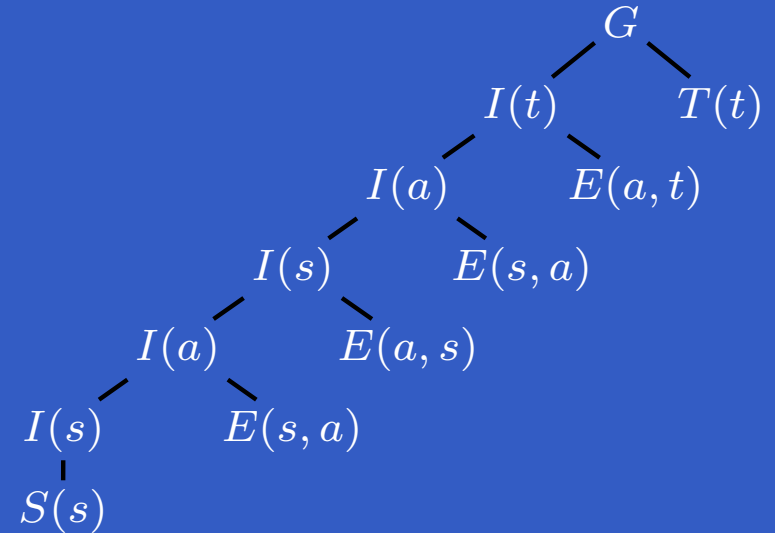
$$I(y) \leftarrow I(x); E(x, y)$$

$$I(x) \leftarrow I(y); E(x, y)$$

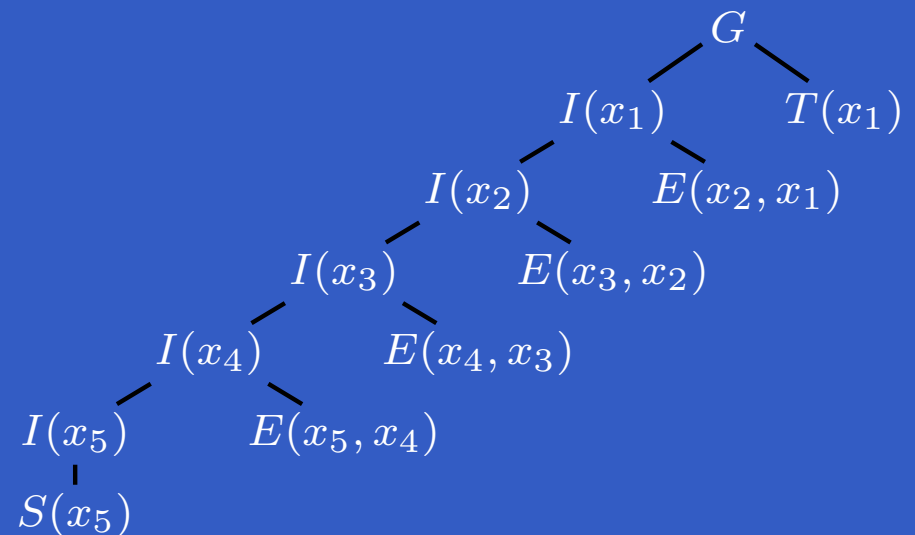
(Rename the vars: $I(y) \leftarrow I(x); E(y, x)$)

$$G \leftarrow I(y); T(y)$$

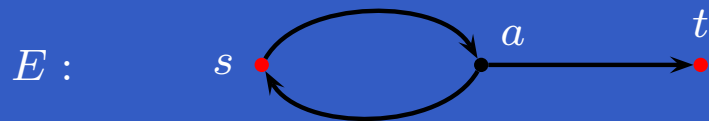
Derivation Path:



Free Derivation Path:



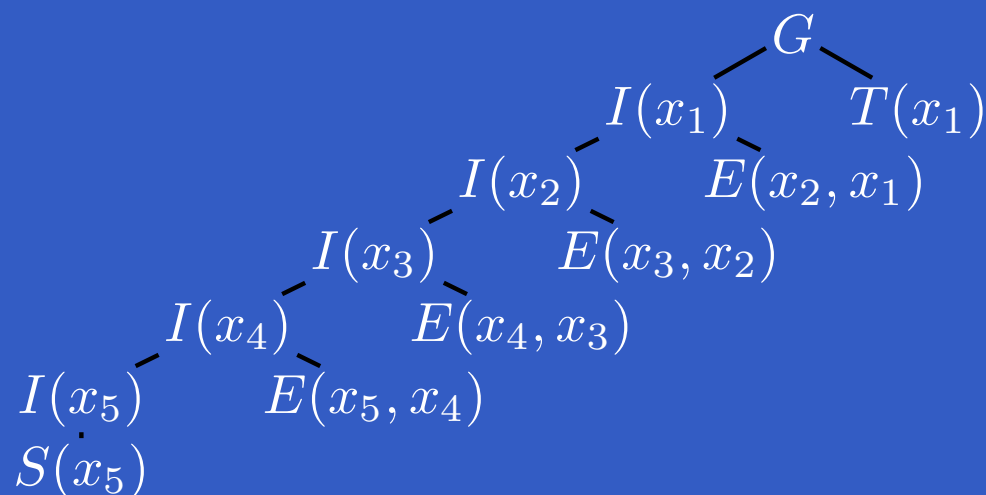
Input Structure:



$$S = \{s\}, T = \{t\}$$

The Free Structure

Free Derivation Path \mathcal{F} :



- The free structure \mathbf{F} is accepted by \mathcal{D}

*

Free Structure \mathbf{F} :

Domain: $F = \{x_1, x_2, x_3, x_4, x_5\}$



$$S^{\mathbf{F}} = \{x_5\}, T^{\mathbf{F}} = \{x_1\}$$

Proof Strategy

- Assume \mathcal{D} works

Proof Strategy

- Assume \mathcal{D} works
- Input: long enough path

Proof Strategy

- Assume \mathcal{D} works
- Input: long enough path
- Abstract away, i.e. take the free derivation path \mathcal{F}

Proof Strategy

- Assume \mathcal{D} works
- Input: long enough path
- Abstract away, i.e. take the free derivation path \mathcal{F}
- Using the symmetricity of \mathcal{D} , “zig-zag” on \mathcal{F} to create a new free derivation path \mathcal{F}' such that:

Proof Strategy

- Assume \mathcal{D} works
- Input: long enough path
- Abstract away, i.e. take the free derivation path \mathcal{F}
- Using the symmetricity of \mathcal{D} , “zig-zag” on \mathcal{F} to create a new free derivation path \mathcal{F}' such that:
 - ◆ In \mathcal{F}' , there is no path from the vertex in S to the vertex in T

Proof Strategy

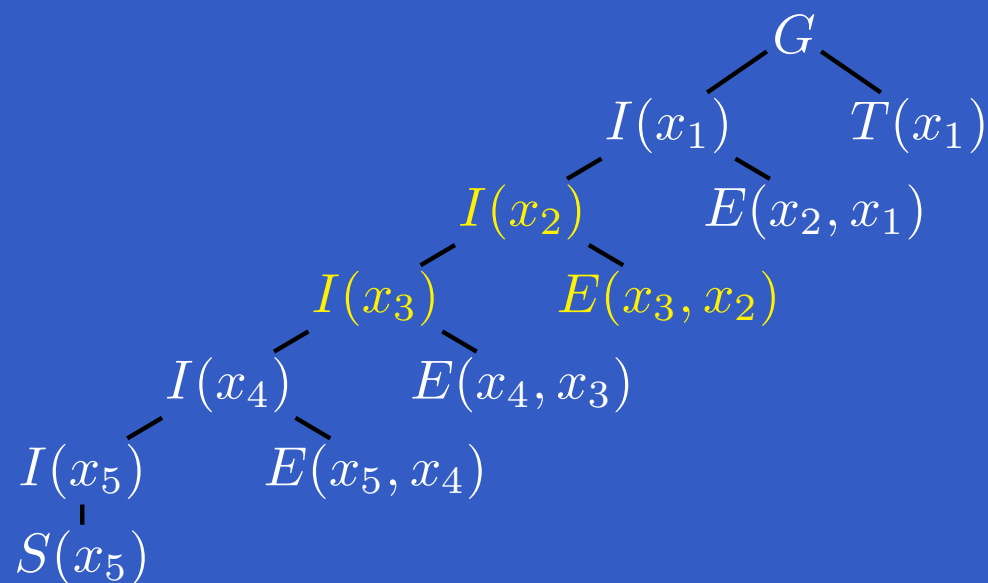
- Assume \mathcal{D} works
- Input: long enough path
- Abstract away, i.e. take the free derivation path \mathcal{F}
- Using the symmetry of \mathcal{D} , “zig-zag” on \mathcal{F} to create a new free derivation path \mathcal{F}' such that:
 - ◆ In \mathcal{F}' , there is no path from the vertex in S to the vertex in T
 - ◆ \mathcal{F}' is a valid derivation path for \mathcal{D} over the free structure of \mathcal{F}'

Proof Strategy

- Assume \mathcal{D} works
- Input: long enough path
- Abstract away, i.e. take the free derivation path \mathcal{F}
- Using the symmetricity of \mathcal{D} , “zig-zag” on \mathcal{F} to create a new free derivation path \mathcal{F}' such that:
 - ◆ In \mathcal{F}' , there is no path from the vertex in S to the vertex in T
 - ◆ \mathcal{F}' is a valid derivation path for \mathcal{D} over the free structure of \mathcal{F}'
- Contradiction

Zig-Zag (Simple Example)

Free Derivation Path \mathcal{F} :



$$I(y) \leftarrow S(y)$$

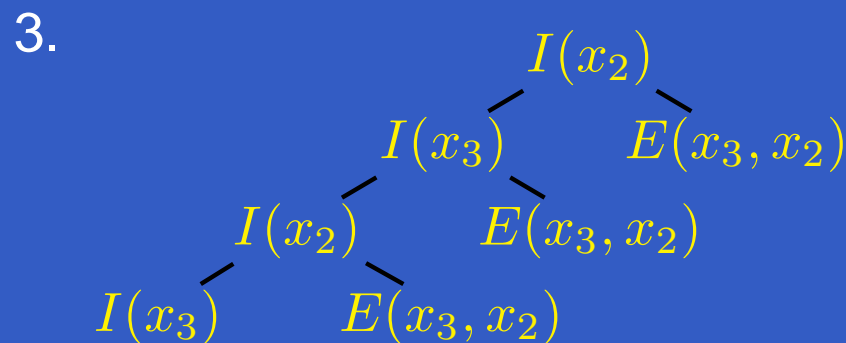
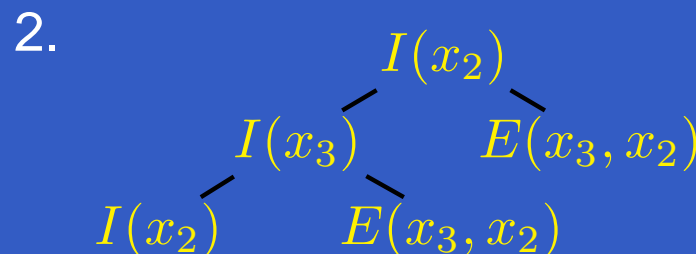
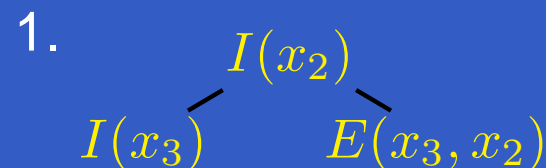
$$I(y) \leftarrow I(x); E(x, y)$$

$$I(x) \leftarrow I(y); E(x, y)$$

$$G \leftarrow I(y); T(y)$$

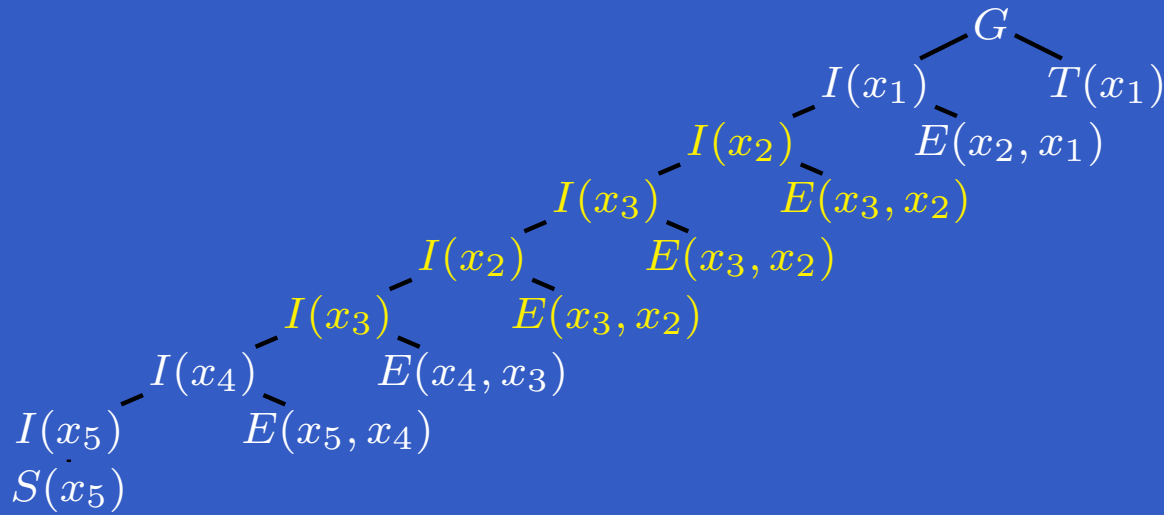
*

Zig-zag (mirror) the yellow segment:

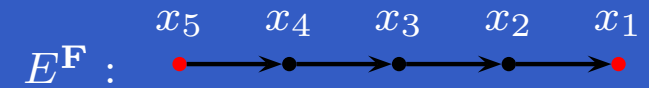


Zig-Zag Continued (Simple Example)

Before renaming the variables in mirrored \mathcal{F}

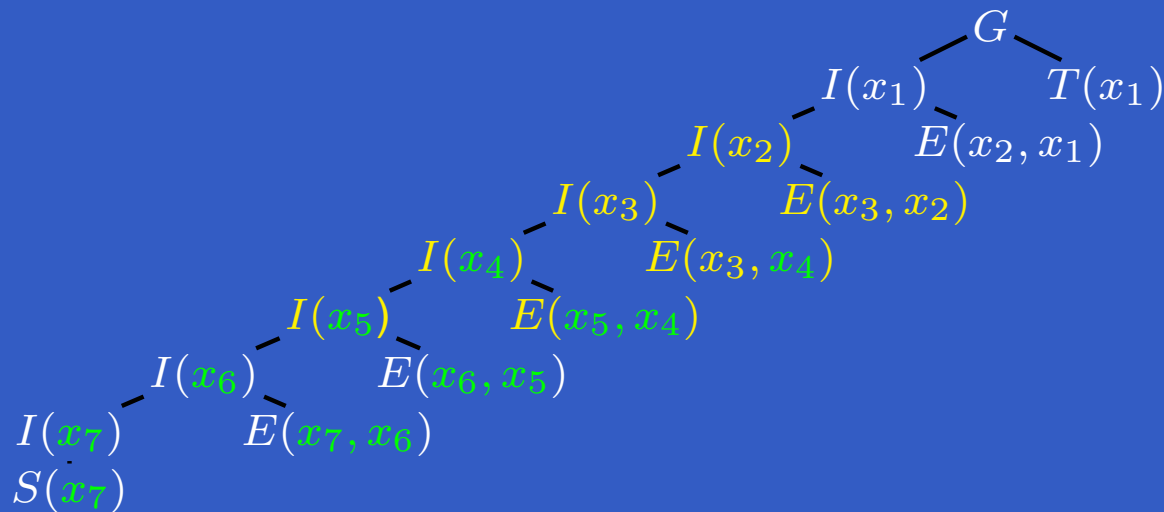


Structure \mathcal{F}

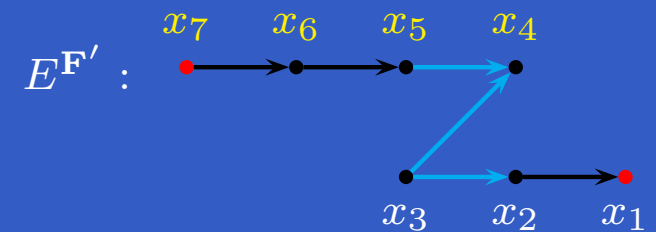


$$S^{\mathcal{F}} = \{x_5\}, T^{\mathcal{F}} = \{x_1\}$$

\mathcal{F}' (variables are renamed):



Structure \mathcal{F}' :



$$S^{\mathcal{F}'} = \{x_7\}, T^{\mathcal{F}'} = \{x_1\}$$

About The General Proof

- Two main complications:

About The General Proof

- Two main complications:
 - ◆ There could be more than one path from the vertex in S to the vertex in T in a free derivation path. **See free structure.**

About The General Proof

- Two main complications:
 - ◆ There could be more than one path from the vertex in S to the vertex in T in a free derivation path. **See free structure.**
 - We disconnect each

About The General Proof

- Two main complications:
 - ◆ There could be more than one path from the vertex in S to the vertex in T in a free derivation path. **See free structure.**
 - We disconnect each
 - Careful, we do not want to create new paths when we disconnect a path

About The General Proof

- Two main complications:
 - ◆ There could be more than one path from the vertex in S to the vertex in T in a free derivation path. **See free structure.**
 - We disconnect each
 - Careful, we do not want to create new paths when we disconnect a path
 - A bit technical

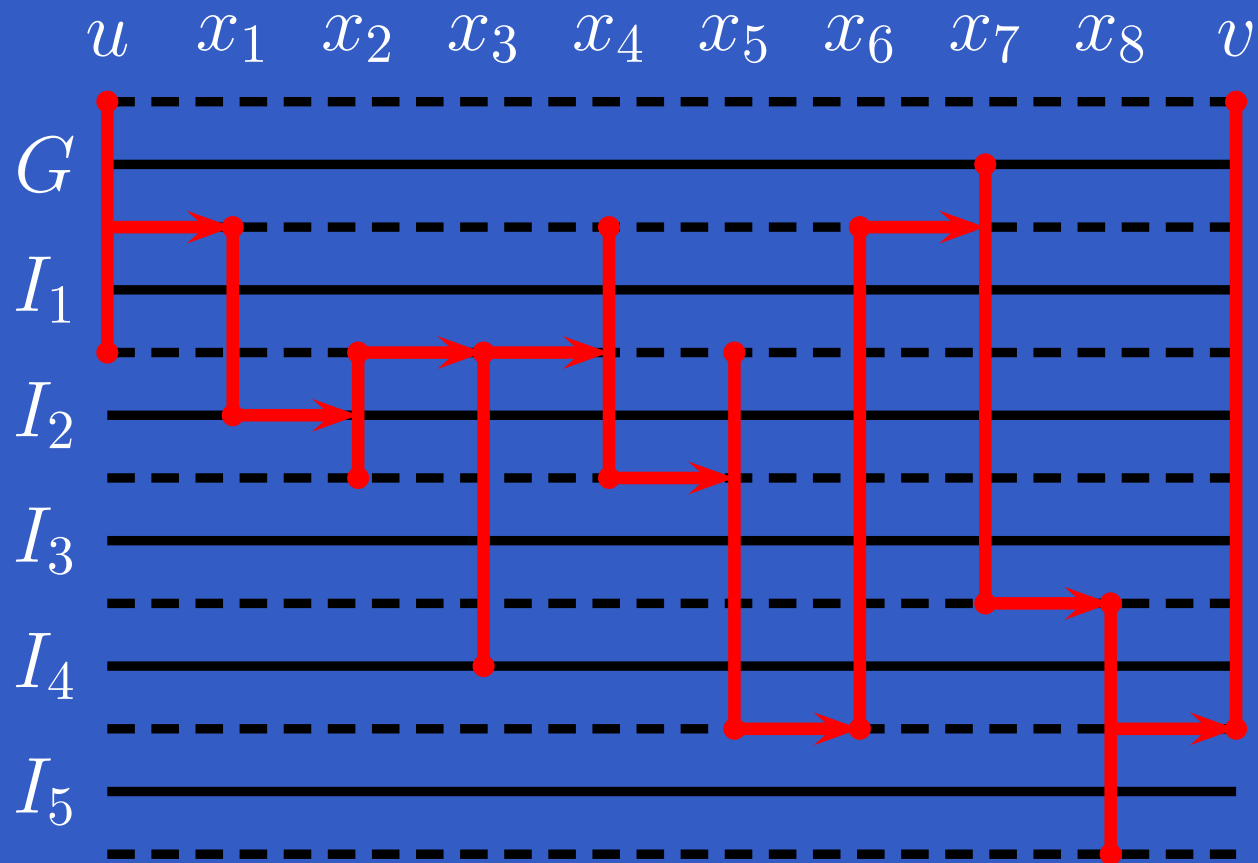
About The General Proof

- Two main complications:
 - ◆ There could be more than one path from the vertex in S to the vertex in T in a free derivation path. **See free structure.**
 - We disconnect each
 - Careful, we do not want to create new paths when we disconnect a path
 - A bit technical
 - ◆ Arity of the IDBs can be arbitrary (but fixed). **See our example program.**

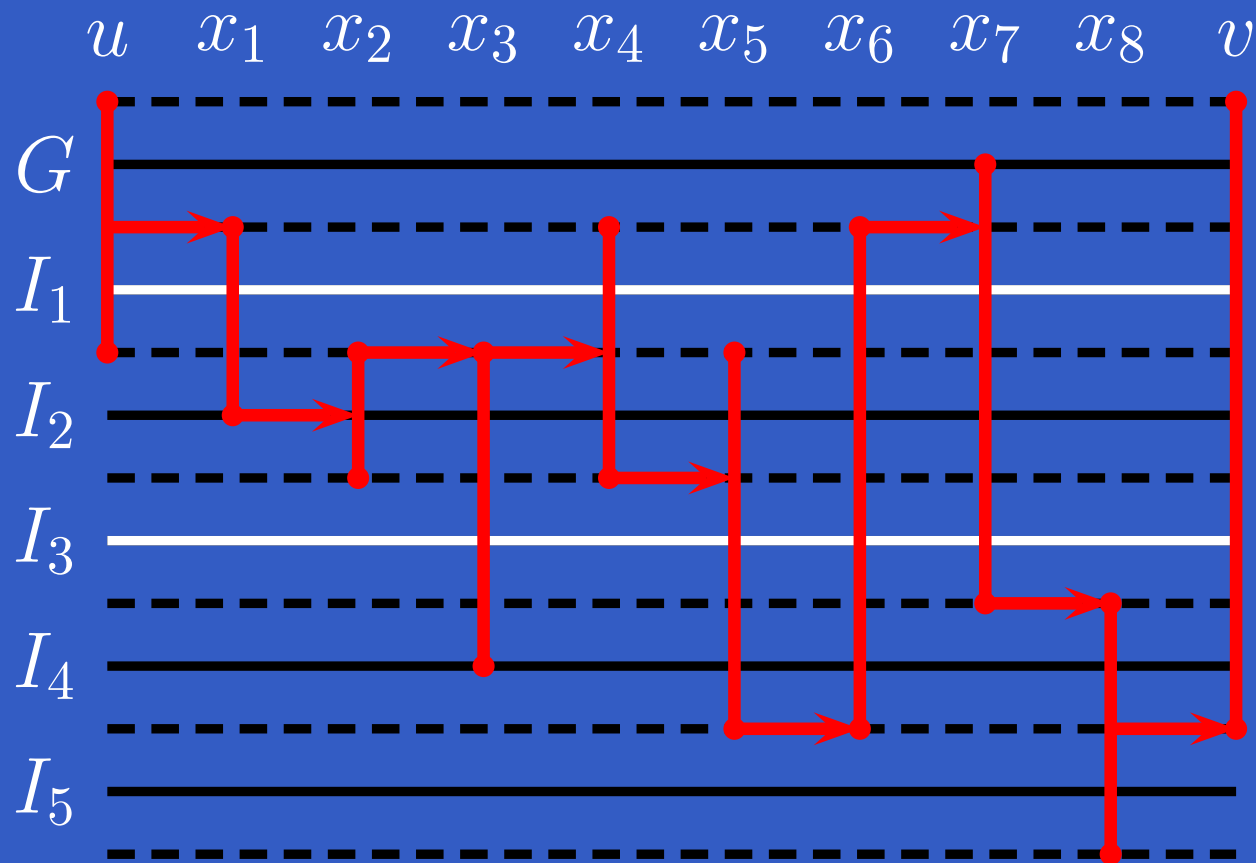
About The General Proof

- Two main complications:
 - ◆ There could be more than one path from the vertex in S to the vertex in T in a free derivation path. **See free structure.**
 - We disconnect each
 - Careful, we do not want to create new paths when we disconnect a path
 - A bit technical
 - ◆ Arity of the IDBs can be arbitrary (but fixed). **See our example program.**
 - We give an intuition how to handle higher arities.

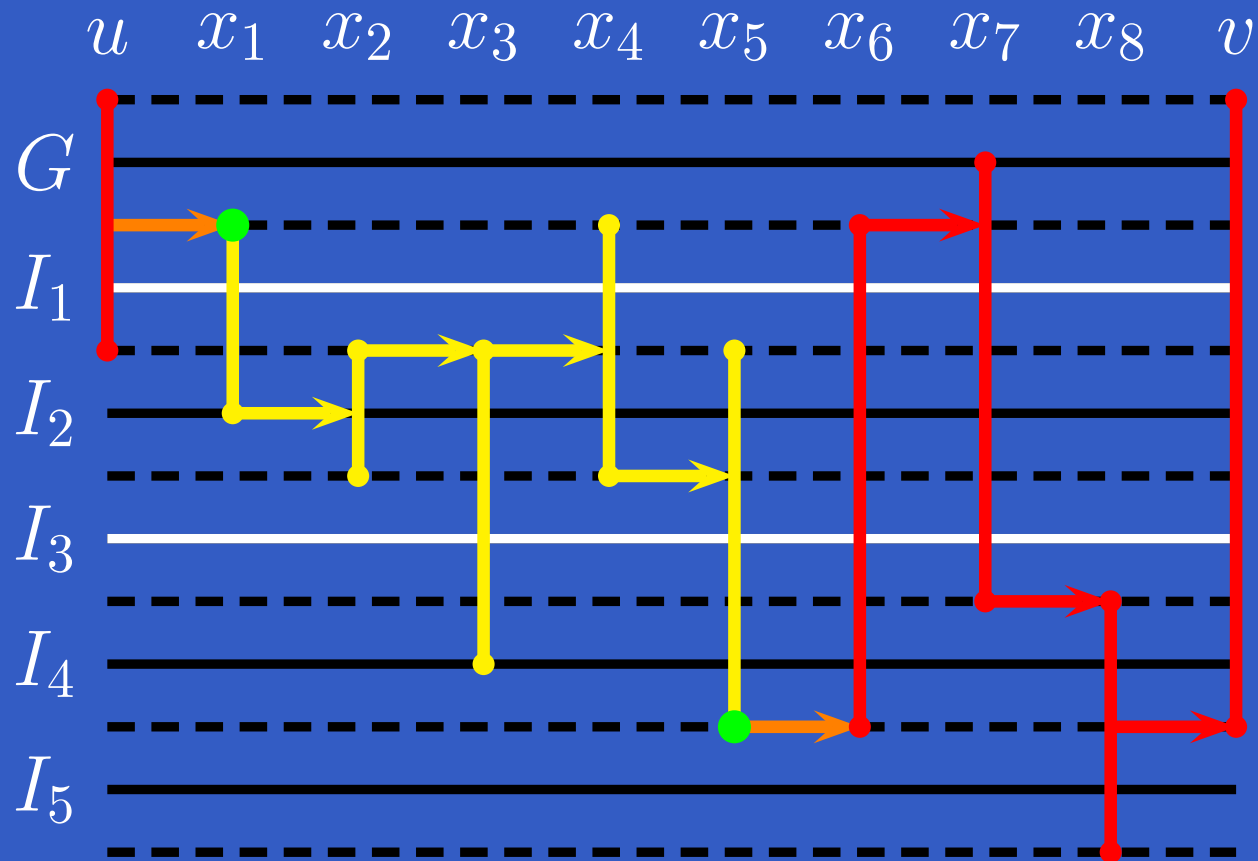
The UV-Path Following Diagram



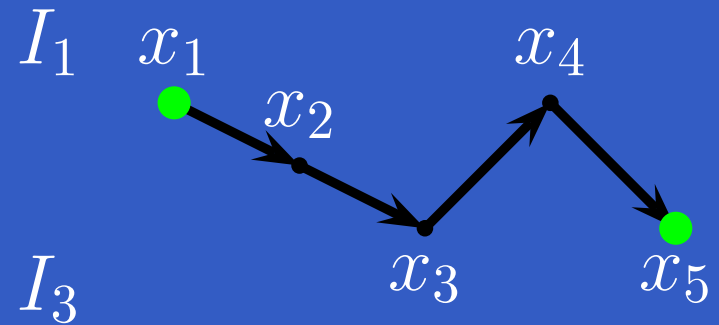
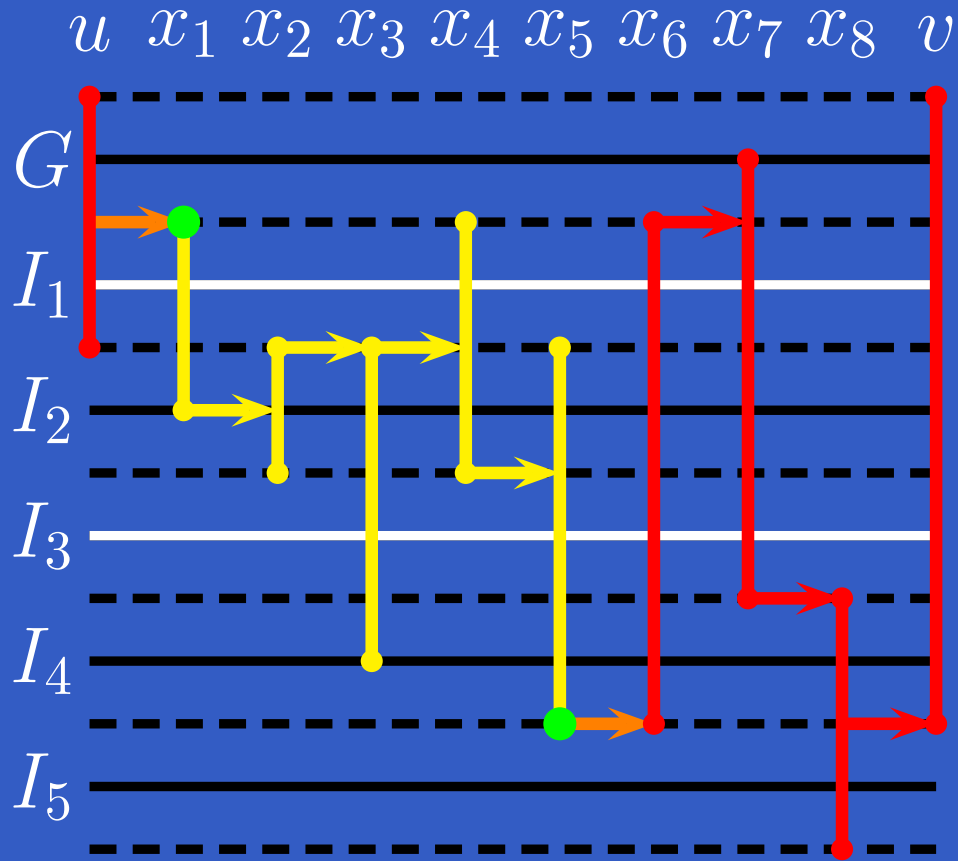
The UV-Path Following Diagram



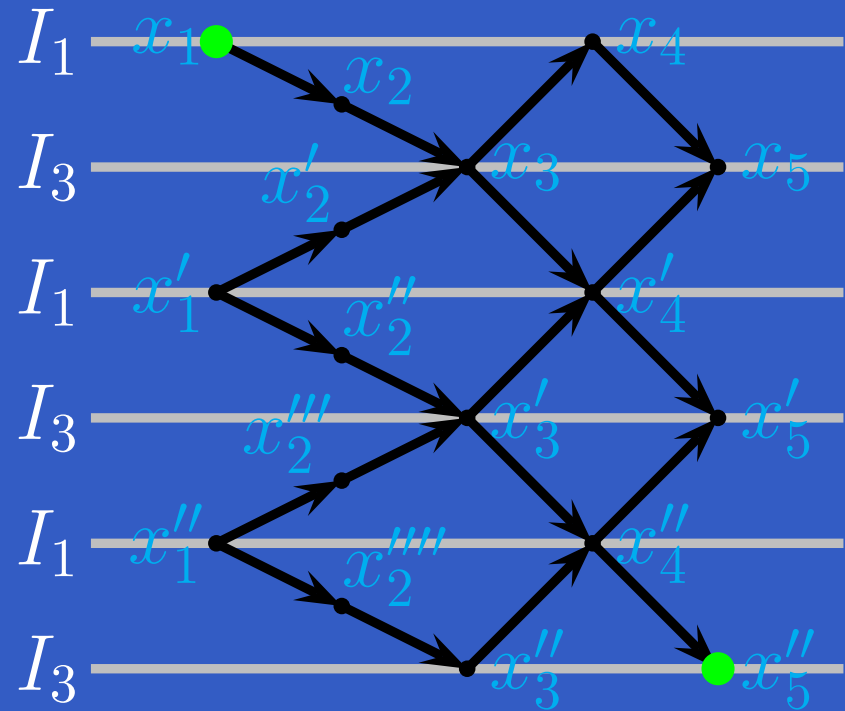
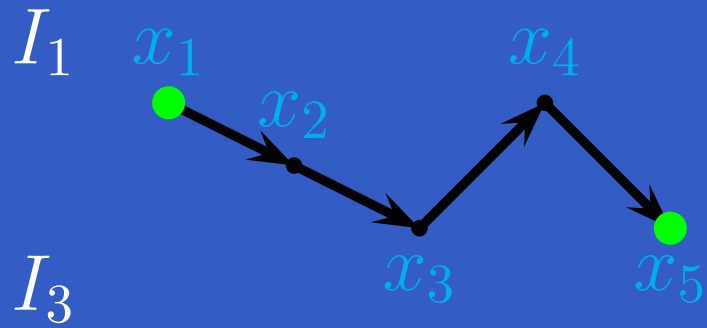
The UV-Path Following Diagram



The UV-Path Following Diagram



Zig-Zag



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions



Questions

