Natural Language Interfaces to Databases

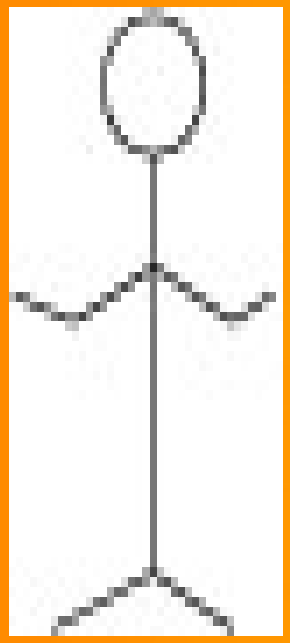
By

Kshitij Bhardwaj
Abhimanyu Rawal
Aman Parnami
Prekshu Ajmera
Lekhraj Meena

Overview

- Introduction
- Motivation
- NLI to Databases?
- Expectations from NLI
- Problems
- Case Studies
- Conclusion

Introduction



User

User : How many students are there in CSE dept?
Comp : There are 50 students in CSE dept.

User : How many teachers?
Comp : Do you want to know the number of teachers in IIT ?

User : How many teachers in CSE dept?
Comp : There are 20 teachers in CSE dept.

Computer with IITB database

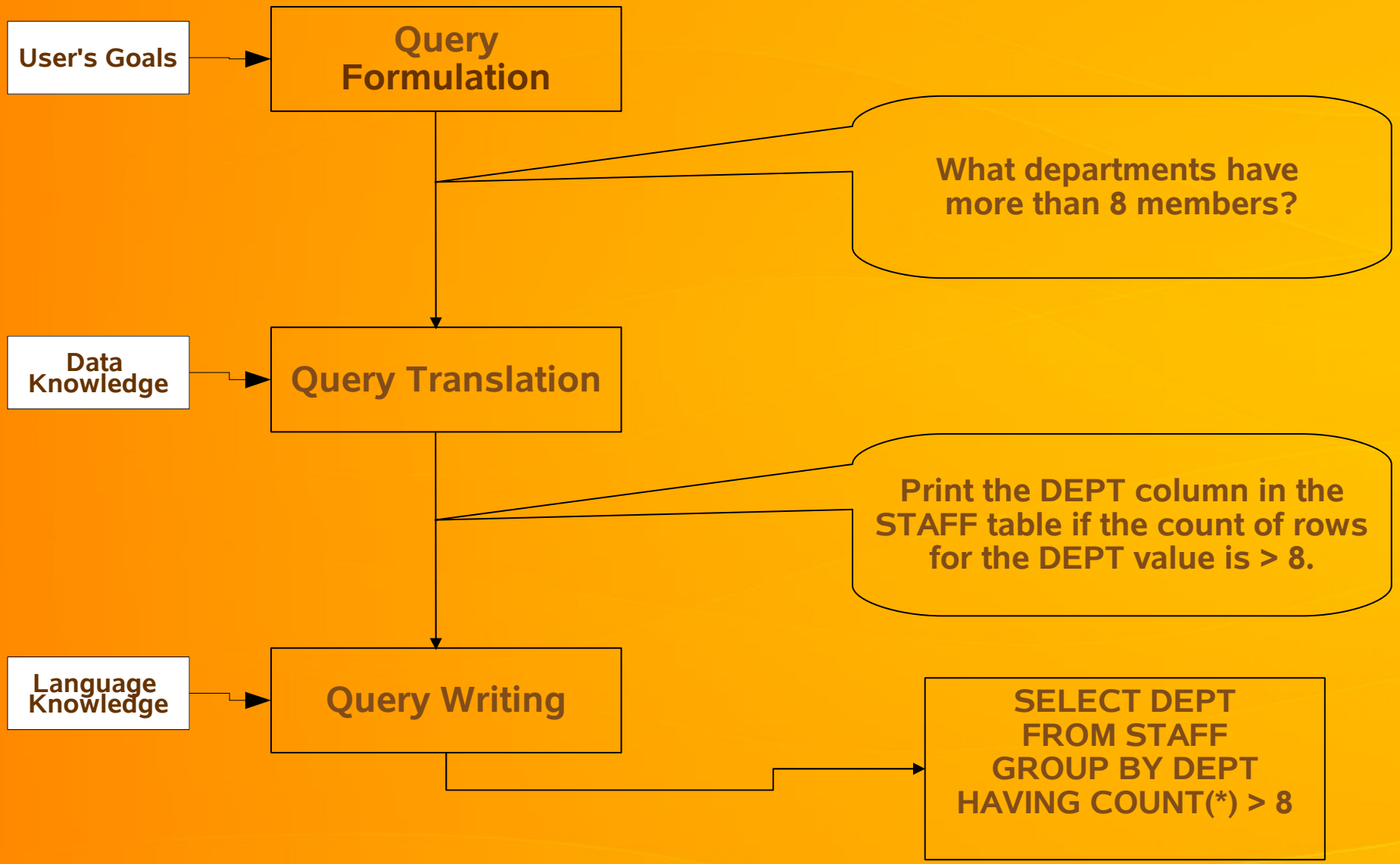
Motivation

- Increasing interaction of non-technical people with databases
- Tremendous use of web browsers, PDAs and cell phones to access information
- Learning query language to interact with a system is inappropriate for many
- Using Natural Language comes naturally!!

Overview

- Motivation
- NLI to Databases?
  - Cognitive model of database query formulation
  - Issues
  - NLI Architecture
- Expectations from NLI
- Problems
- Case Studies
- Conclusion

Cognitive Model of Query Writing



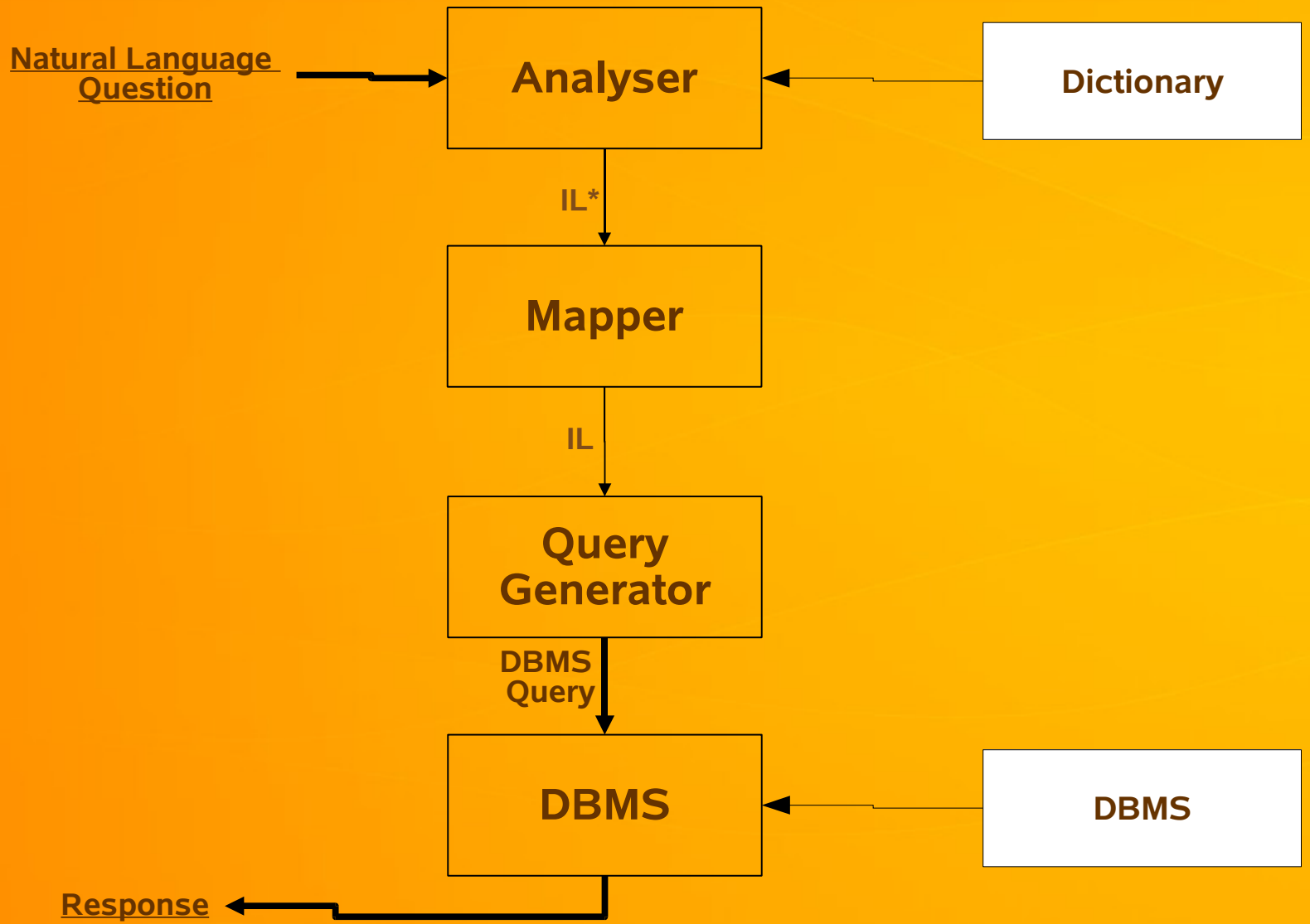
Issues

- 3 broad categories are:
  - Knowledge acquisition and representation
    - Requirements of human m/c dialogue and interaction
    - Capture and formalise information efficiently
    - Incorporate knowledge into framework
  - Language Processing techniques
    - Assign structure and interpretation to queries
  - Database issues
    - Formulation of correct structured query
    - Thorough understanding of DBMS structure

NLI Architecture

- 3 main components of NLID system:
  - Analyser : parsing of i/p into tokens
  - Mapper : converts o/p of Analyser into database relation and attribute names
  - Query Generator : generates database query

NLI Architecture



* Intermediate language(ex BURG)

Expectations from a NLI

- Developer's View :
  - Minimal application dependency
  - Least effort configuration to new DBMS
- User's View :
  - Fast response time
  - Answer most queries
  - Ask for clarifications if required
- Others :
  - Handling of non-standard questions
  - Portability to different m/c

Problems

- Application Definition Problems:
  - Recognising values to be put in database
  - Deciding number and types of record
- Language Problems
  - Tense and time
    - eg. How far did the Fox travel yesterday? (yesterday as an interval over which an event extends)
    - Who was the officer of the day yesterday? (yesterday as a point in a sequence of days)
  - Ellipsis and anaphora
  - Yes/No questions
    - eg. Has Rakesh been interviewed?
    - 'no' may come due to lack of knowledge also

Problems contd...

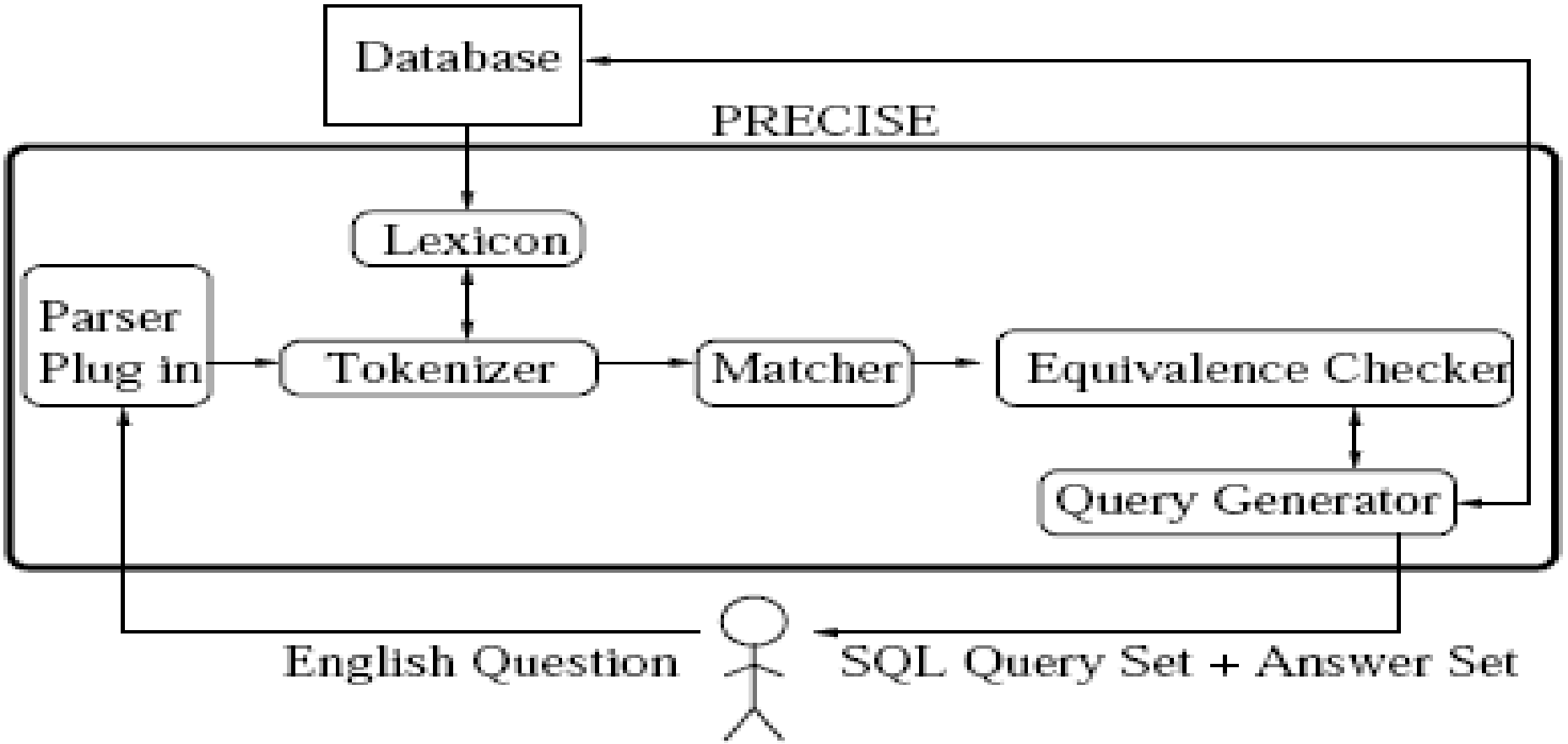
- Conjunctions : scope of conjunctions
- Negation : interpretation is difficult
- Others :
  - Syntactic Ambiguity : multiple valid parses of same query
    - eg. The man drove down the street in a car
  - Semantic Ambiguity : determining the intended referent in database
    - eg. Who advises users in numbers 2510?
  - Vagueness : the absence of detail that would normally be explicit in formal database queries
    - eg. Q. Which students passed CS345? (vague)
      Q. Which students got a passng grade in CS345?

Case Studies

PRECISE

- Based on following principle:
  - Guarantees correctness of output
  - Accept if something is not understood
- Is transportable to arbitrary databases
- Graph based
- Answers 80% of semantically tractable questions
- Recognizes other unanswerable 20% questions

PRECISE : System Architecture



Semantically Tractable Questions

- Tokenization contains distinct tokens
- Atleast one token matches a wh-value (e.g:what, where etc.)
- A valid mapping from set of tokens to database elements(attributes, values, relations)

Tokenizer

- I/P : Natural Language Question
- A *token* is a set of word stems that matches a database element
  - For ex : {require, experience} matches 'Required Experience' --> Database Attribute
- More than one token-attribute mappings are possible
  - For ex : {need, experience} will also match 'Required Experience'
- Stems each word of the question and looks up the lexicon

Mapper(Matcher)

- I/P : Tokens
- Maps set of tokens to set of database elements

Algorithm

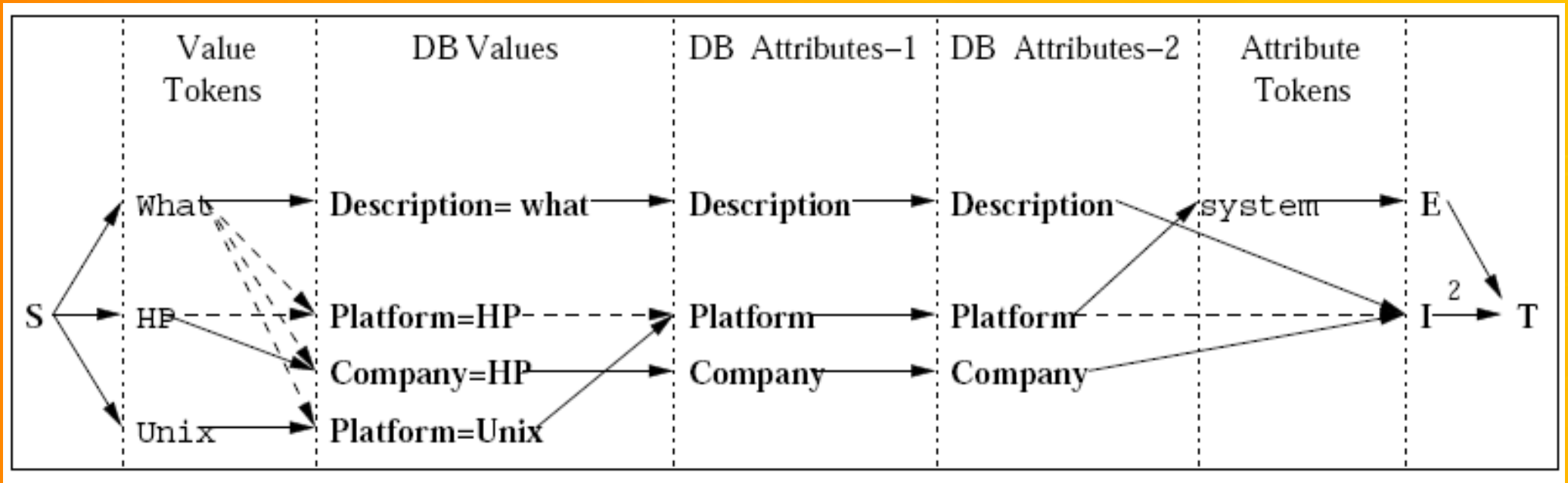
1. Construct attribute-value graph
2. Runs *max-flow* algorithm on graph
3. Returns **unambiguous** mapping

Mapper : Example

Ques : What are the HP jobs on a UNIX System?

tokenization

{What, HP, jobs, UNIX, System}



Attribute-value graph created by PRECISE for above given question and tokens

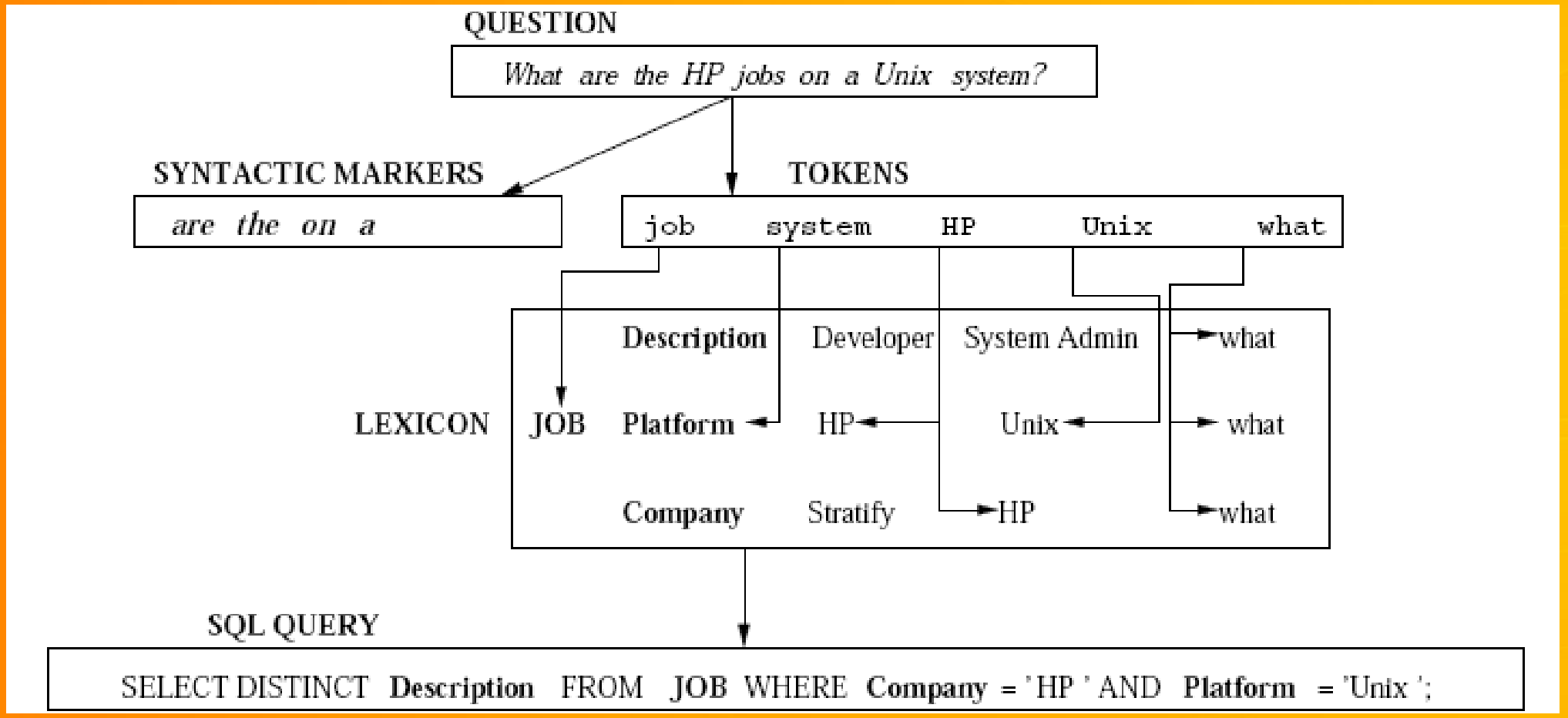
Query Generator

- I/P : Database elements selected by Mapper

SELECT <DB elements paired with wh-words>
WHERE <conjunction of attributes & their values>
FROM <relation names for attributes in WHERE>

DBMS Query Structure

Example



In this example database contains a single relation, JOB, with attributes Description, Platform and Company

LIFER

- Language Interface Facility with Ellipsis and Recursion
- General facility for creating and maintaining linguistic interfaces
- Composed of 2 basic parts
  - Set of interactive language specification functions
  - Parser
- Other Accessories
  - Spelling correction, paraphrasing, incomplete inputs

LADDER

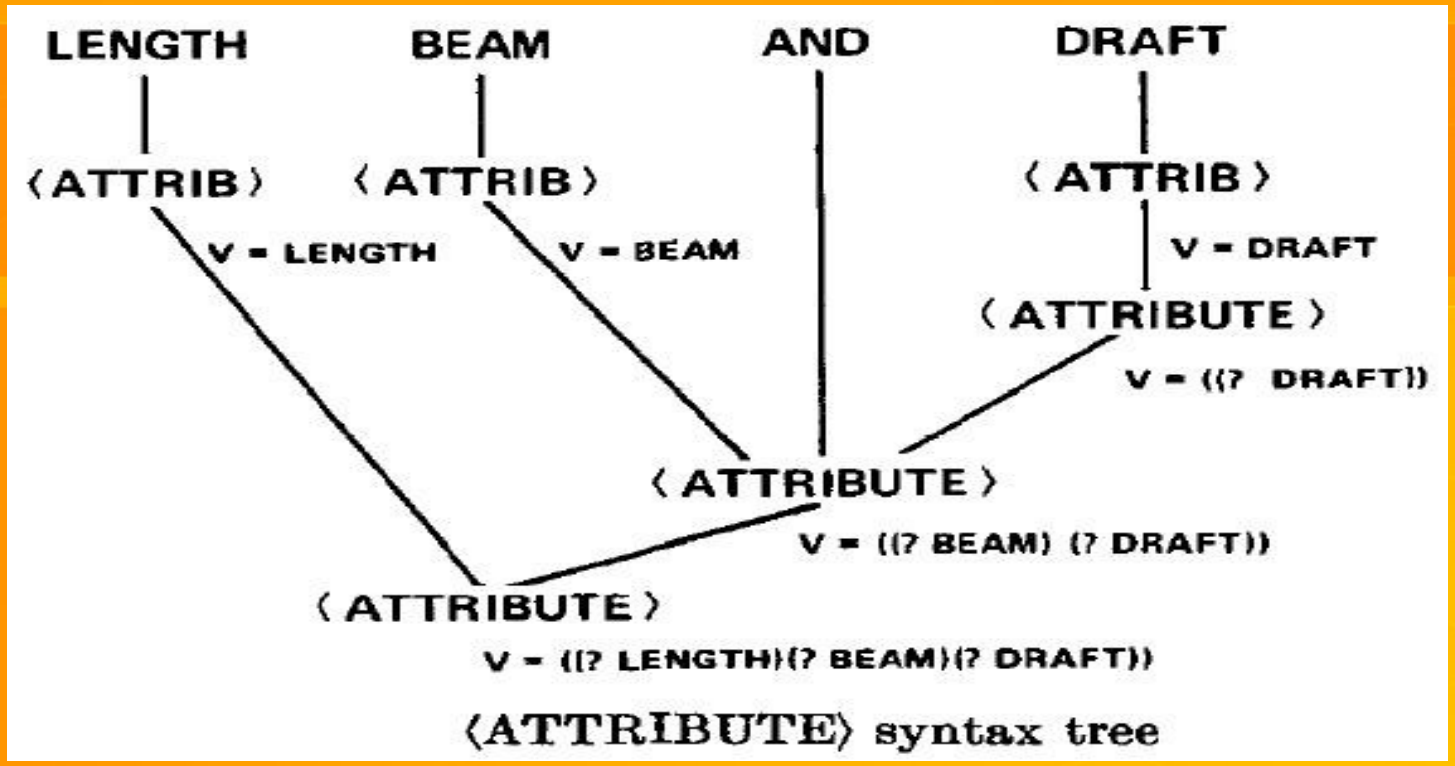
- Language Access to Distributed Data with Error Recovery
- Prototype system developed by SRI
- Automated procedure of technicians
- Developed as a management aid to navy decision makers
- Composed of 3 components :
  - INLAND
  - IDA
  - FAM

INLAND

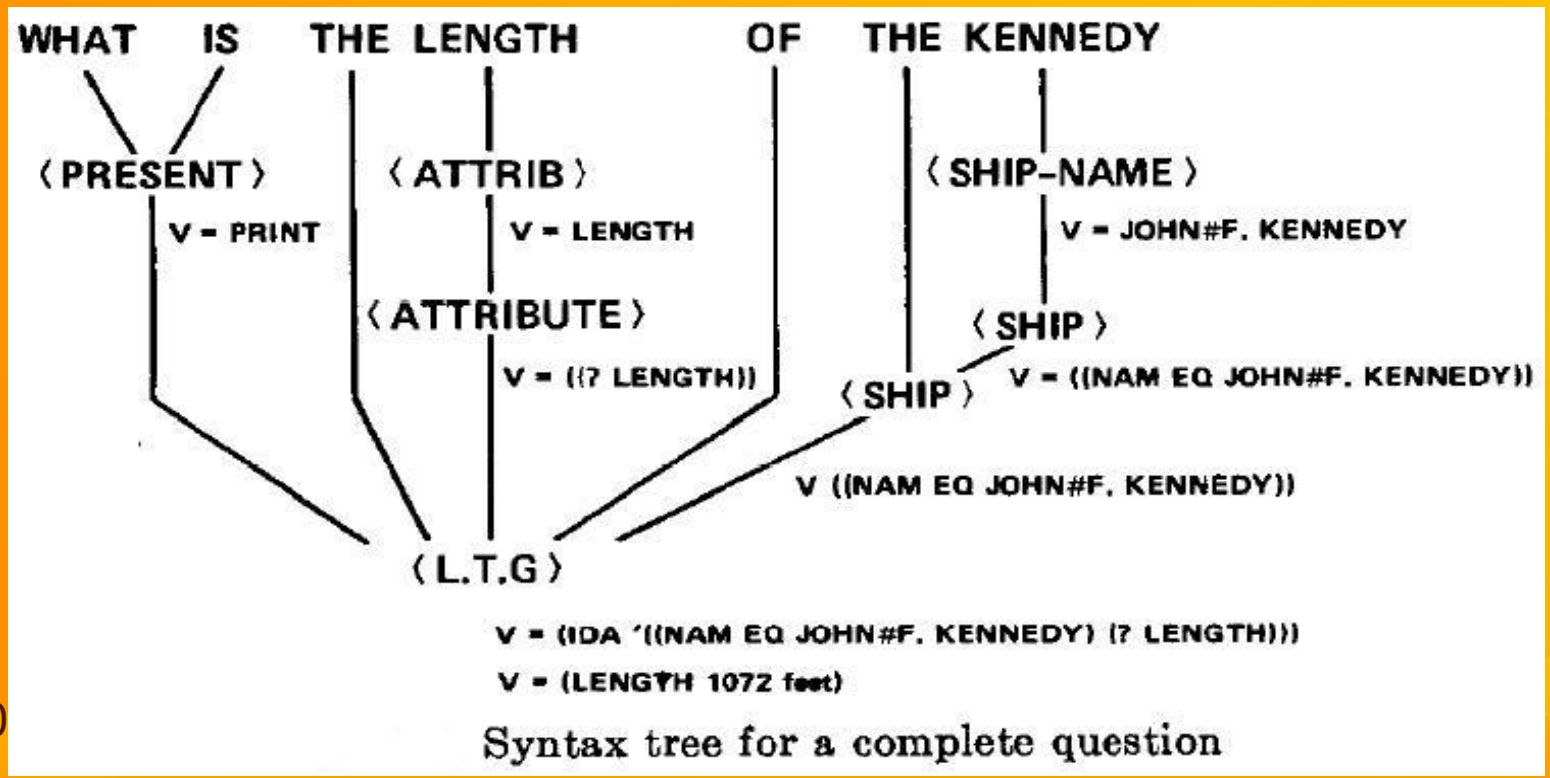
- Informal Natural Language Access to Navy Data, accepts restricted subset of NL
- Incorporates special purpose LIFER semantic grammar
- <L.T.G> (LIFER Top Grammar) highest level meta-symbol of grammar
- Parses (top-down) natural language to give LISP expression(patterns) which is fed as input to IDA
- It is NLI to IDA

INLAND contd...

- Example pattern :
- <L.T.G> --> <PRESENT> THE <ATTRIBUTE> OF <SHIP>
- The LISP expression for above will be :
  - (IDA (APPEND <SHIP> <ATTRIBUTE>))
  - Here <SHIP> and <ATTRIBUTE> values will be obtained by the parser while parsing instance



⟨ATTRIBUTE⟩ syntax tree



Syntax tree for a complete question

IDA

- Intelligent Data Access
- Presents a structure-free view of a distributed database
- Needs to know remote DBMS
- Processes the Lisp query and breaks it down against the entire VLDB into a sequence of queries against individual files on DBMS
- IDA composes answers to the original query

FAM

- File Access Manager
- Maps generic file names onto specific file names on specific computers on specific sites
- Initiates network connections
- Opens files
- Monitors for certain errors
- Returns answer to single-gram queries to IDA

INLAND Limitations

- LIFER allows only CFGs to be defined, english language could be outside CFG too
- YES/NO questions
- No Assertions – designed for retrieval
- LIFER does not deal with Syntactic Ambiguity directly – accepts first successful analysis
  - eg. - Is A nearer to B than C
  - Deep Parsing
- INLAND cannot read articles and expand database

Applications

- Railway reservation and enquiry machine
- Customer care services
- All query systems

Conclusion

- NLIs if developed are the most natural way to interact with DBMS.
- All the issues mentioned should be resolved for this technique to succeed.
- Incorporating flexibility to adapt different DBMS is needed for widespread usage.
- It is the need of the hour to integrate the benefits of different systems evolved till now.

References

- S. Jerrold Kaplan. Designing a Portable Natural Language Database Query System. *In ACM transactions on Database Systems* 9(1), pages1-19,1984 194
- W. C. Ogden, "Implications of a Cognitive Model of Database Query: Comparison of a Natural Language, a Formal Language and a Direct Manipulation Interface," *ACM SIGCHI Bulletin,* vol. 18, pp. 51-54, 1985.
- M. Templeton and J. Burger. Problems in Natural Language Interface to DBMS with Examples from EUFID. *In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California,* pages 3--16, 1983
- Ana-Marie Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. *In Proceedings of the conference on Intelligent User Interfaces*, 2003
- G. Hendrix, E. Sacerdoti, D. Sagalowicz, and J. Slocum. Developing a natural language interface to complex data . *In ACM transactions on Database Systems* 3(2), pages105–147, 1978.
- B. Grosz, D. Appelt, P. Martin, and F. Pereira. TEAM: An Experiment in the Design of Transportable Natural Language Interfaces. *In Artificial Intelligence* 32, pages 173–243, 1987.

Thank You

Questions ?