Instructional semantics

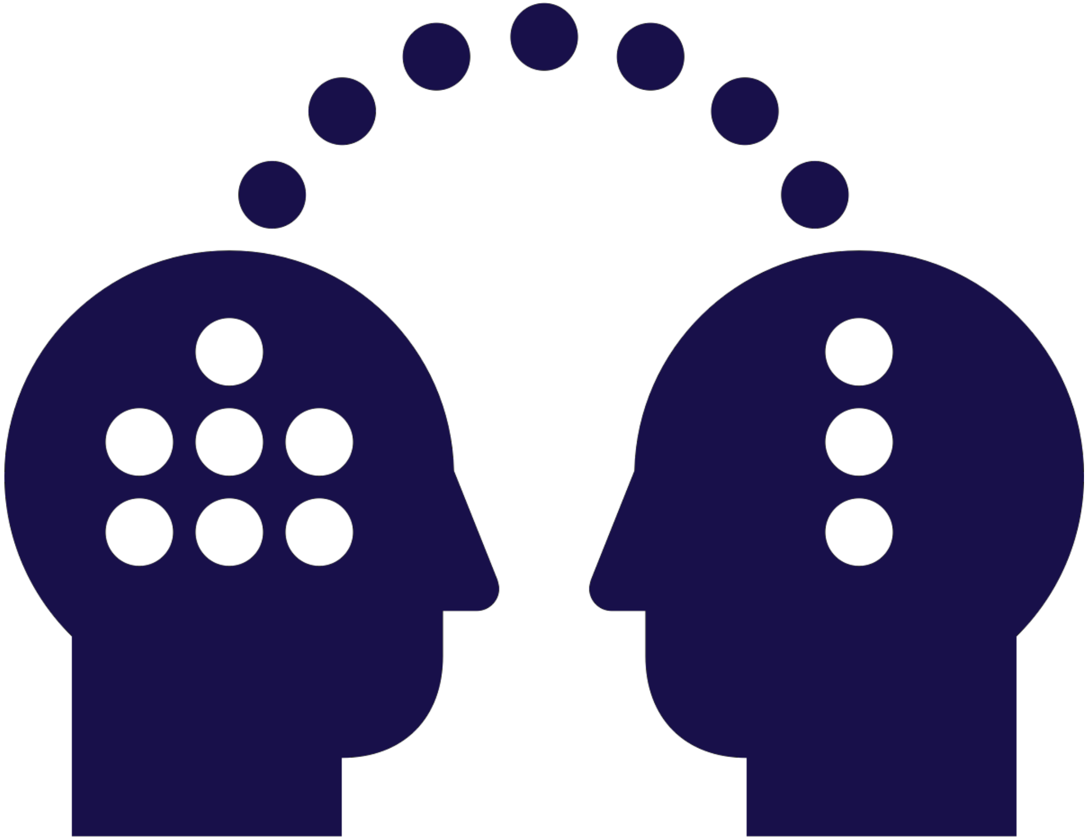
How to write a book in SQL

Fourth Workshop
"Computational linguistics and language science"
Moscow, HSE, April 2019

Ivan Rygaev
Laboratory of Computational Linguistics
Institute for Information Transmission Problems, Russian Academy of Sciences
irygaev@gmail.com

Communication

- The main purpose of language is communication
  - Information transfer from one mental database to another



Knowledge

- Mental database
  - Stores our knowledge of the objects we got familiar with
  - Can be represented in the form of a semantic graph or a set of logical propositions
- Prevailing semantic theories
  - Relate the meaning of a sentence to the knowledge itself
  - What is transferred is just a part of it
  - A book is knowledge stated piece by piece

Knowledge consistency

- Problem
  - New knowledge must be related to the old one
  - We do not have direct access to the hearer's MDB
  - We cannot put new knowledge in the right cell directly
  - We can only ask the hearer to do so (give her instructions)
- *The child woke up*
  - Put 'woke up' in the cell of the child
  - Formally:
    find x: child(x)
    update x: wake_up(x)

Related work

- Davies & Isard 1972
  - Utterances as programs
  - Two-step processing: compilation and executions
  - Understanding a sentences vs carrying it out
- Heim 1982
  - Metaphor of file keeping
  - *“For every indefinite, start a new card; for every definite, update a suitable old card.”*

Instructional semantics

- The meaning of a sentence
  - A sequence of instructions to update the hearer’s MDB
  - A book is a script which creates the knowledge
- Each instruction
  - Has a certain propositional content
  - Is related to a particular mental referent
- Instruction types
  - find – to identify an existing mental referent
  - create – to create a new mental referent
  - update – to add new information about a mental referent

Examples

- (What did Peter do?) Peter built a house
  - find x: named(x, ‘Peter’)
  - create y: house(y)
  - update x: build(x, y)

x
named(x, ‘Peter’)
build(x, y)

y
house(y)
build(x, y)

- (Who built the house?) The house was built by Peter
  - find y: house(y)
  - find z: build(z, y)
  - find x: named(x, ‘Peter’)
  - update z: z = x

Two-step process

- Compilation step
  - Understanding the sentence
  - Building an instructional representation of it
- Execution step
  - Carrying out a sentence (a voluntary process)
  - Applying instructions to the mental database
  - An updated mental representation has a truth value
- Presupposition failure
  - If ‘find’ instruction fails then ‘update’ cannot be performed
  - No updated mental state – no truth value

Information structure

- Definiteness
  - ‘find’ instruction corresponds to a definite/known referent
  - ‘create’ – to an indefinite/new referent
- Givenness
  - ‘find’ instruction contains given information
  - ‘create’ and ‘update’ – new information
- Topic/comment
  - The topic is a mental referent of ‘update’ instruction
  - It is usually also associated with ‘find’ instruction
  - Content of ‘update’ instruction constitutes the comment

More instructions

- Questions
  - 'request' – to request information about a certain mental referent. The main instruction for questions.
  - *Who built the house?*
  - find y: house(y)
    request x: build(x, y)
- Parenthetical constructions
  - 'secondary update' – to update a mental referent which is not the main topic of the sentence
  - *Peter, <u>a friend of mine</u>, built a house*

Interaction with context

- Context independence
  - Instructional representation has no references to context
  - No variables refer directly to any real or mental referent
  - They will get their assignments only on the second step
- Context relevance
  - Each set of instructions is only relevant in a certain context
  - They provide an answer to a particular question
  - Each sentence answering the same question (ideally) has the same set of instructions

One set of sentences

- *Who built the house?*
  - *The house was built by PETER*
  - *PETER built the house*
  - *It is PETER who built the house*
- Instructional representation
  - find y: house(y)
    find z: build(z, y)
    find x: named(x, ‘Peter’)
    update z: z = x

Another set of sentences

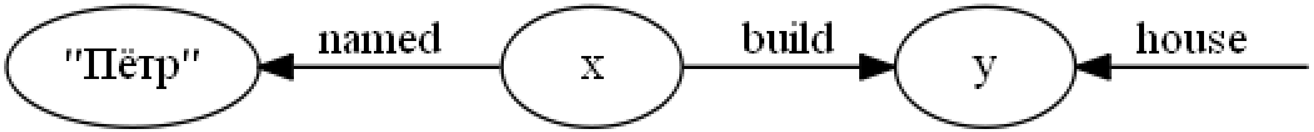
- *What did Peter build?*
  - *Peter built THE HOUSE*
  - *THE HOUSE was built by Peter*
  - *What Peter built is THE HOUSE*
- Instructional representation
  - find x: named(x, ‘Peter’)
    find w: build(x, w)
    find y: house(y)
    update w: w = y
- Same situation
  - Different question answered -> different instructions

Interlingua

- Instructional meaning
  - Common for all context-relevant paraphrases
- Good candidate for semantic interlingua
  - Completely language-independent
  - Stripped of the original syntax and lexicon
  - But captures the information structure (context-relevance)
  - Should generate translations which are communicatively adequate in the same context

Knowledge representation

- Speaker knowledge:
  - ∃x, y (named(x, ‘Peter), build(x, y), house(y))
- Semantic graph:



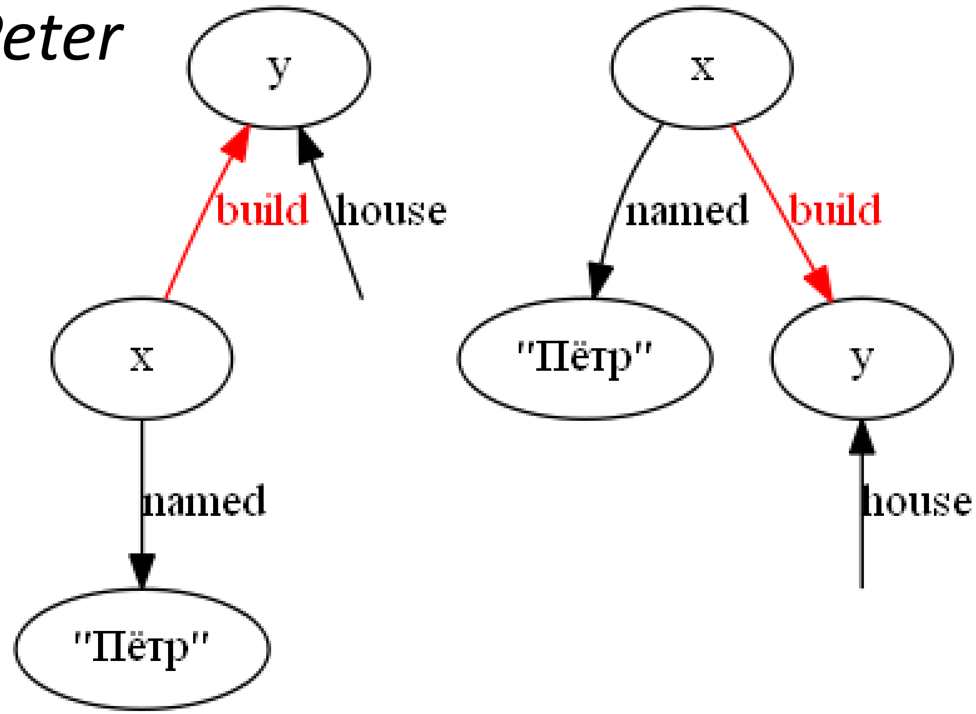
  - Nodes correspond to referents (variables)
  - Arcs correspond to predicates
- Not a tree-like structure
  - No definite root
  - Can contain loops

Arborization

- The question
  - How does a syntactic tree arise out of a non-tree-like semantic graph?
- Instructional semantics is the answer
  - Instructions split the graph into small subgraphs
  - Each instruction has a head node, hence it is a subtree
  - Connecting subtrees generates the whole tree
  - The root node is the topic (the head node of ‘update’)
- Semantic tree
  - It is then lexicalized and linearized

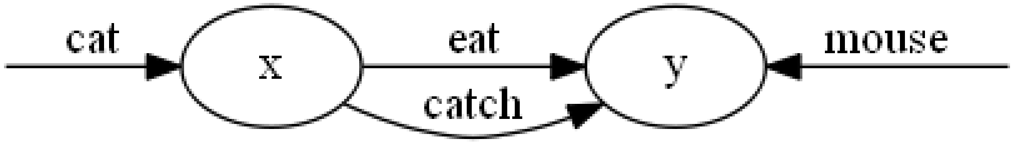
Arborization example

- Notation
  - Arrow directions show semantic dependencies
  - Node positions – communicative/syntactic dependencies
- *The house was built by Peter*
  - find y: house(y)
    find x: named(x, ‘Peter’)
    update y: build(x, y)
- *Peter built the house*
  - find x: named(x, ‘Peter’)
    find y: house(y)
    update x: build(x, y)



Loops

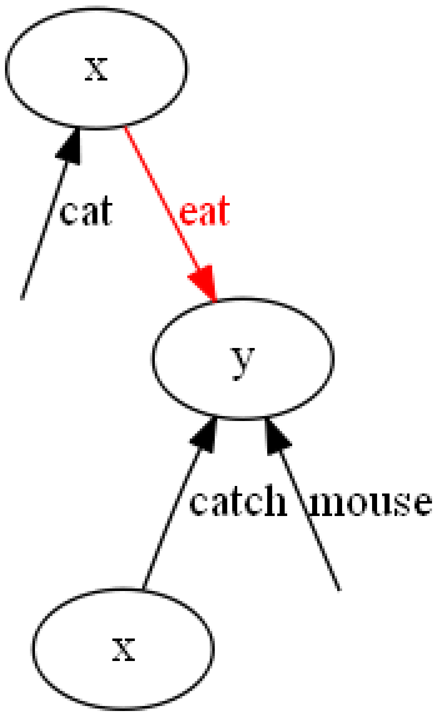
- Phrases with loops:
  - *The cat is eating the mouse which it caught*
  - *The cat caught the mouse which it is eating*
- Speaker knowledge:
  - ∃x, y (cat(x), eat(x, y), catch(x, y), mouse(y))
- Semantic graph:



  - Splitting into instructions helps to remove the loops (by duplication of the nodes)

Removing loops

- *The cat is eating the mouse which it caught*
  - find x: cat(x)
  - find y: mouse(y), catch(x, y)
  - update x: eat(x, y)
- Constituents structure
  - It is arcs (predicates) which are linguistically realized
  - Empty nodes – pronouns
  - [[cat$_i$]] [eat [[mouse] [[PRO$_i$] catch]]]
  - [the [cat$_i$]] [is eating [the [mouse] [[it$_i$]caught]]]



Island constraints

- Island
  - A syntactic construction which contains an element that cannot be extracted out of it
- Example
  - *John loves [the sister who lives in Paris]*
  - find x: named(x, ‘John’)
    find z: named(z, ‘Paris’)
    find y: sister(y, x), live(y, z)
    update x: love(x, y)

Island constraint violation

- Ungrammatical sentence
  - ******Where*** *does John love [the sister who lives in ___]?*
  - find x: named(x, 'John')
    find y: sister(y, x), lives(y, z) // z is undefined
    request z: loves(x, y) // z in not mentioned though requested
- Grammatical sentence with the same meaning
  - *Where does the sister live which John loves?*
  - find x: named(x, 'John')
    find y: sister(y, x), love(x, y)
    request z: live(y, z)

Islands explanation

- Each instruction is an island
  - An ungrammatical structure arises when we try to use the same propositional content within two different instructions (with different head referents)
  - The resulting set of instructions is not sensible and cannot be executed sequentially
  - Extraction can occur only within the content of one instruction

Future work

- Future work
  - Provide psycholinguistic evidence for the adequacy of the model
  - Define derivation rules, how an instructional representation can be built compositionally from the syntax of the sentence
  - Define exact rules of instruction execution
  - Cover other linguistic phenomena (plurals, quantifiers, etc.)
  - Build computational tools for text analysis and synthesis in terms of instructional semantics
  - Explore whether instruction clash can account for all cases of island constraints

Conclusions

- Instructional semantics
  - Reflects speaker intentions and resembles psycholinguistic processes of language production and comprehension
  - Captures the information structure (context-relevance) and serves well as an interlingua for translation
  - Explains how a syntactic tree arises out of a semantic graph of speaker knowledge
  - Can explain the existence of syntactic islands
  - Produces an updated mental representation, which captures the logical form, has a model-theoretic interpretation and can be used for inferences

Thank you for you attention!
Questions?