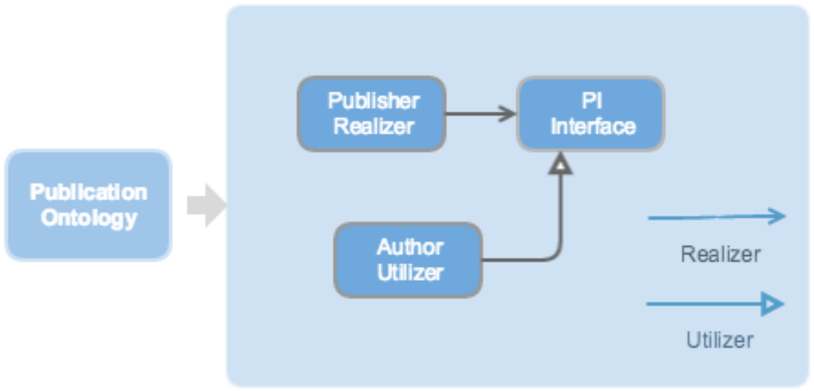
Distributed Ontology Reasoning

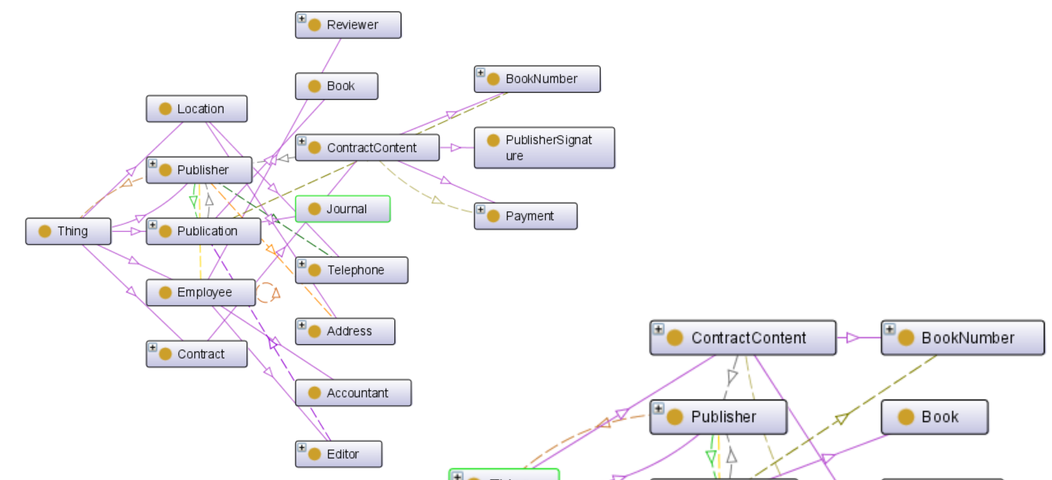
Li Ji and Weichang Du
Faculty of Computer Science, University of New Brunswick

Distributed Ontology

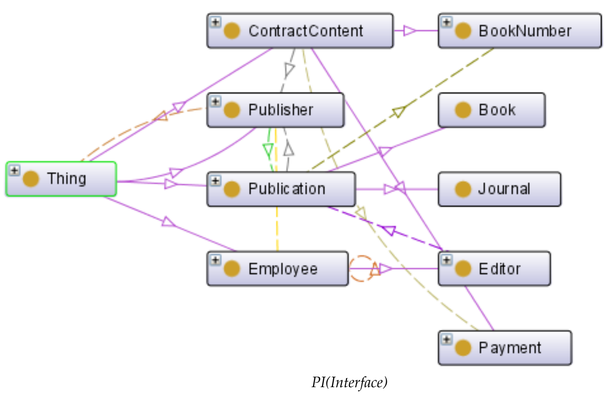
A monolithic ontology is decomposed to several modular ontologies stored on different local computers, which will generate distributed ontology.



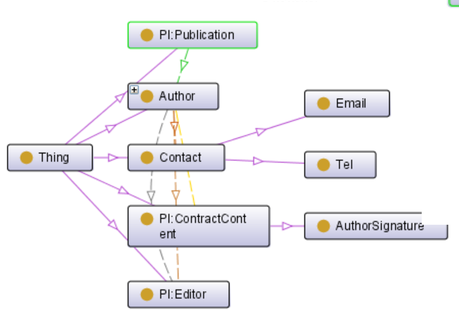
Publication ontology can decompose to *Publisher*, *Author* and *PI*. Particularly, *Publisher* realizes concepts and roles of interface *PI* and *Author* utilizes knowledge from PI.



Publisher



PI(Interface)



Author

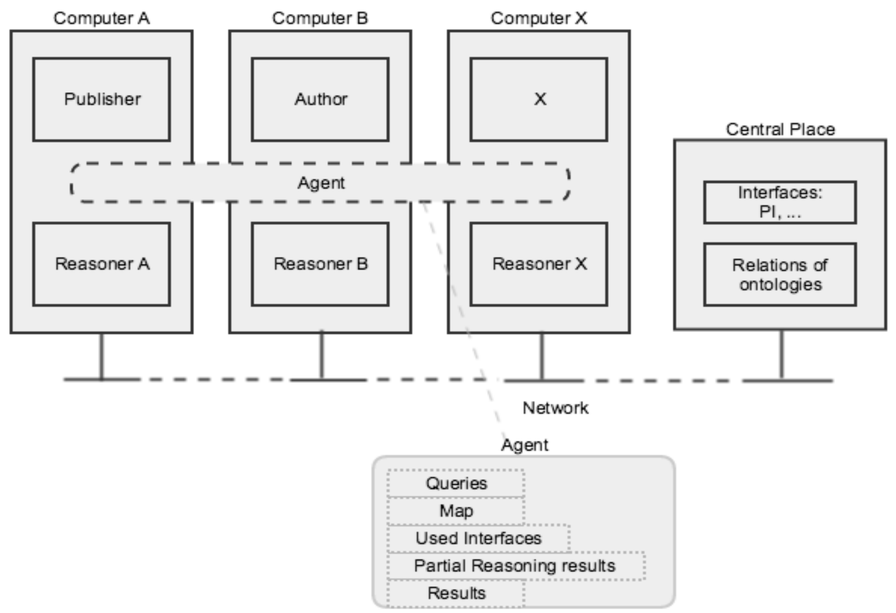
In above image, the concepts (PI:Publication, PI:ContractContent and PI:Editor) are not defined by *Author*. *Author* just uses the concepts from *PI* by its roles. For example, *Author* defines role hasEditor without Editor. But it uses PI:Editor as the ranges of hasEditor.

Distributed Ontology Reasoning

1)Author: Author_Signature ⊑ Publisher: Contract
It represents that the concept Author_Signature of *Author* is subsumed by Contract of *Publisher*.
2) workWith (Author: Author, Publisher: Reviewer)
3) ∃ Publisher: employ.(Publisher: Accountant ⊓ Publisher: Editor) ⊓ ∃Author:hasEmail.Author:Email
4) ∃Author:hasEditor.(Publisher: Employee ⊓ Publisher: Editor) ⊔ ∃ Publisher: publish.(Publisher: Book ⊓ Publisher: Journal)

The reasoning process of the last statement:
Firstly it needs to reason *Publisher* to get the result of Employee ⊓ Editor. Secondly, when we get the result (Editor), the statement will be changed to Author: hasEditor.Publisher: Editor. Then, this statement will be reasoned in *Author*. After that, the result is not inconsistent. Meanwhile, Because this statement belongs to disjunction. Finally, the whole statement is not inconsistent.

Agent-Based Distributed Ontology Reasoning Platform



A reasoning process: A user needs to find instances about Contract in *Publisher*, *PI* and *Author*.
At First, the agent finds Contract and its subsets in *Publisher*. Next, it stores all concepts' instances from *Publisher*. Meanwhile, compared these concepts with *PI*, the agent labels related concepts of *PI*. Then, carrying information, the agent travels to *Author* and gets the instances according to the label. Finally, the agent returns all instances to the user.