# Knowledge Sharing System
## 3. Semantic Web
### 3.2 Description Logic


武田英明　Hideaki Takeda

takeda@nii.ac.jp

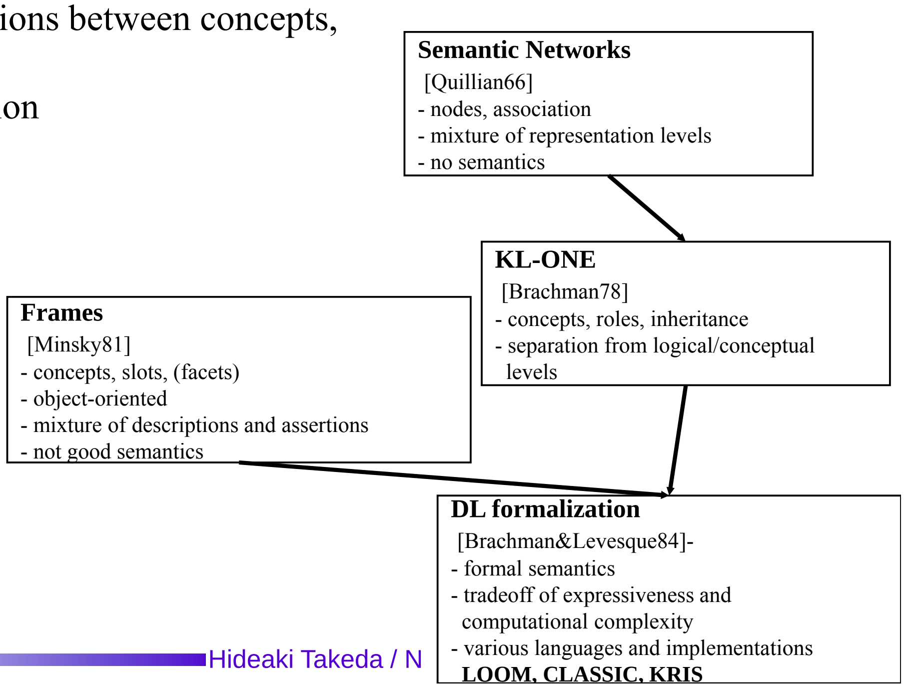
国立情報学研究所

National Institute of Informatics

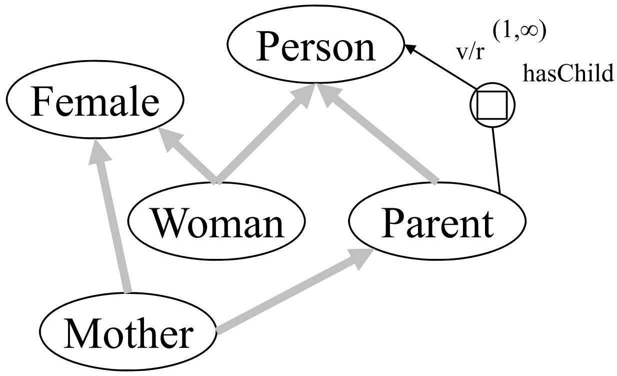



# Description Logic

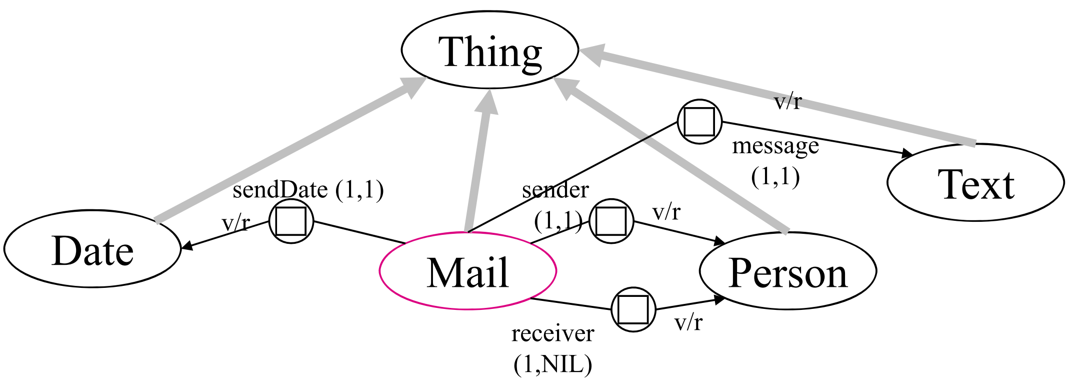
- What is Description Logic?
  - Representation for structured knowledge level
    - Concepts, relations between concepts, inheritance
  - Logical formalization
- History



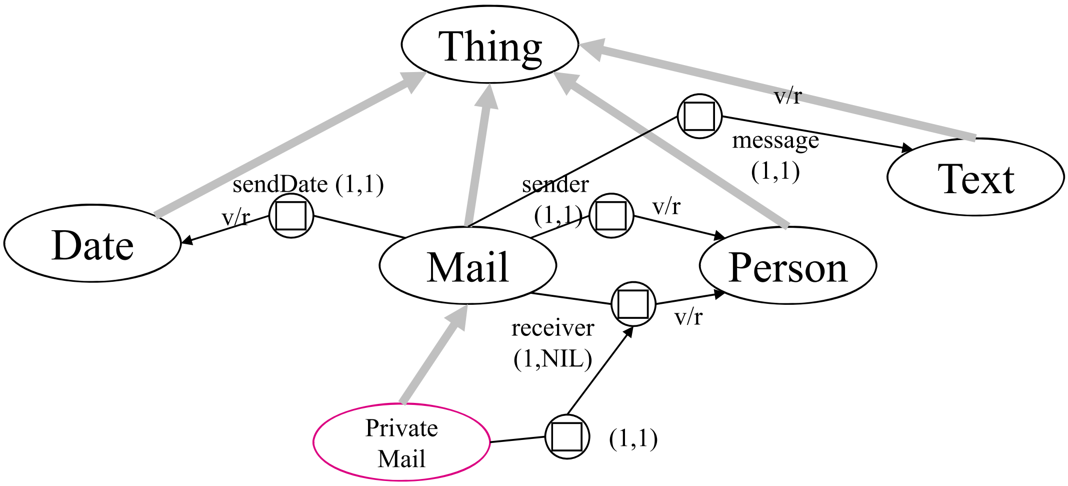
# Concept



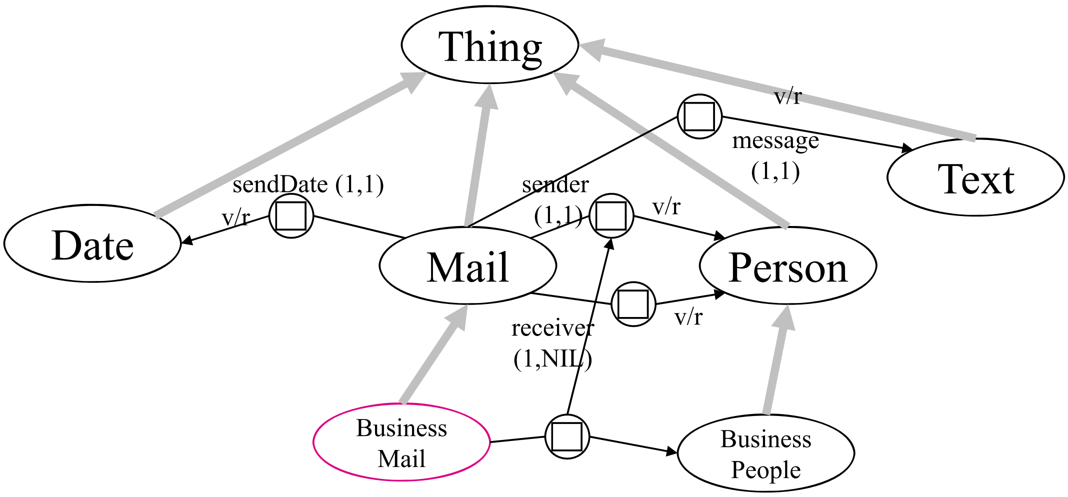


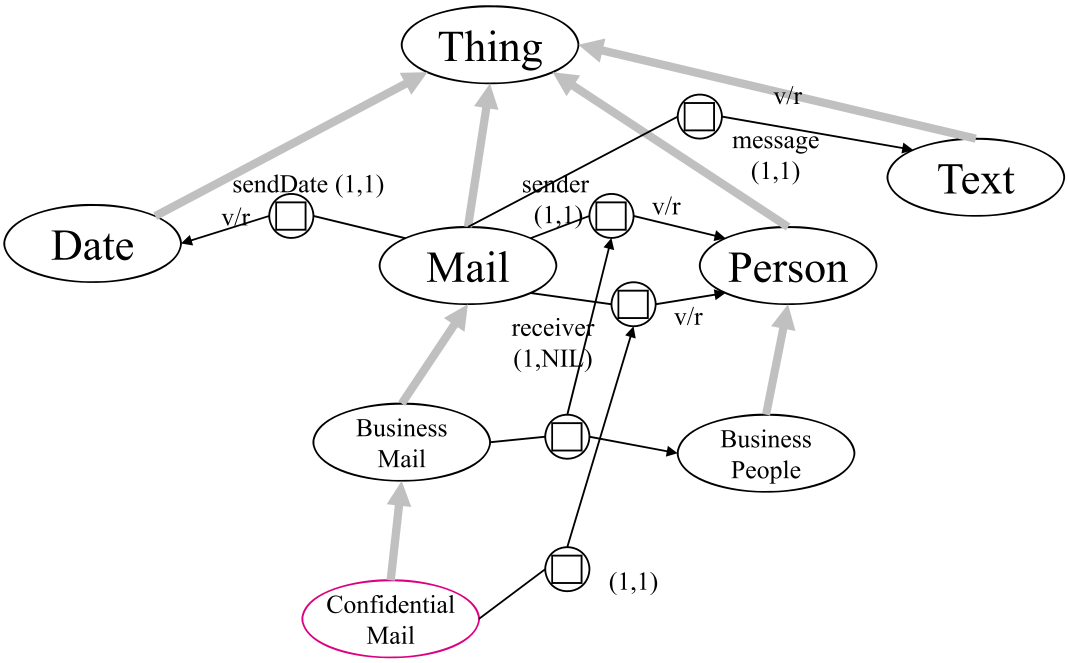




$$\text{Mail} \subseteq \text{Thing} \cap \forall\text{sendDate.Date} \cap \forall\text{sender.Person} \cap \forall\text{receiver.Persor}$$


Thing
v/r
message
(1,1)
Text
sendDate (1,1)
v/r
Date
sender
(1,1)
v/r
Mail
Person
receiver
(1,NIL)
v/r
Private
Mail
(1,1)




Thing
v/r
message
(1,1)
Text
sendDate (1,1)
v/r
Date
sender
(1,1)
v/r
Mail
Person
receiver
(1,NIL)
v/r
Business
Mail
Business
People






# Elements of Description Logic

- Concepts: entities and classes
    - Person
  - *Unary predicates in FOL*
    - {x | Person(x)}, λx.Person(x)
- Roles: properties and relations
    - haschild
  - *2-ary predicates in FOL*
    - {x, y | hasChild(x, y)}
- Constructors for concept expression: conjunction(∩), union(∪)
    - Person ∩ ∃hasChild.Female
    - {x | Person(x) ∧ ∃y.haschild(x, y) ∧ Female(y)}
- Individuals: instances of concepts, co-reference to objects in the world
  - Ex, Takeda, s1234

# Constructors for concept expression

$FL^-$ $AL^*$

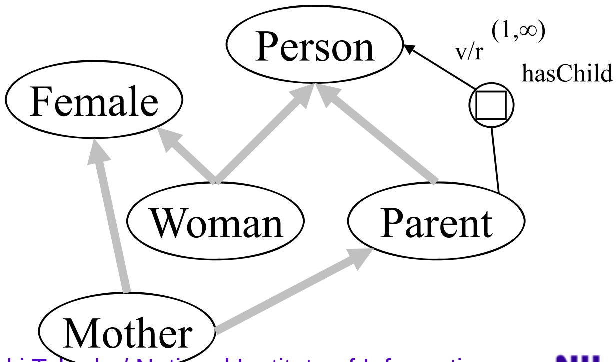
| Constructors | Syntax | Semantics |
|---|---|---|
| **Concept** | **C** | $C^I \subseteq \Delta^I$ |
| **Role name** | **R** | $R^I \subseteq \Delta^I \times \Delta^I$ |
| **Conjunction** | $C \cap D$ | $C^I \cap D^I$ |
| **Value restriction** | $\forall R.C$ | $\{x \in \Delta^I \mid \forall y.(x,y) \in R^I \Rightarrow y \in C^I\}$ |
| **Existential quantification** | $\exists R$ | $\{x \in \Delta^I \mid \exists y.(x,y) \in R^I\}$ |
| **Negation** | $\neg C$ | $\Delta^I \setminus C^I$ |
| **Top** | $\top$ | $\Delta^I$ |
| **Bottom** | $\bot$ | $\emptyset$ |
| **Disjunction** | $C \cup D$ | $C^I \cup D^I$ |
| **Existential restriction** | $\exists R.C$ | $\{x \in \Delta^I \mid \exists y.(x,y) \in R^I \wedge y \in C^I\}$ |
| **Number restriction** | $(\geq n\ R)$ | $\{x \in \Delta^I \mid \lvert\{y \mid y.(x,y) \in R^I\}\rvert \geq n\}$ |
| **Collection of individuals** | $\{a_1, a_2, \ldots\}$ | $\{a_1^I, a_2^I, \ldots\}$ |

# Semantics

- Interpretation *I* consists of the domain of discourse $\Delta^I$ (non empty set) and interpretation function $^I$
  - $^I$ maps
    - Concept C to $C^I \subseteq \Delta^I$
    - Role R to $R^I \subseteq \Delta^I \times \Delta^I$
- A model for C is an interpretation where $C^I$ is not empty
- A concept C is satisfiable if it has a model for C

# TBox & ABox

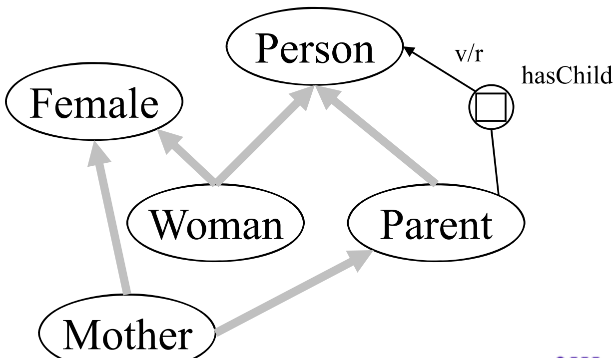
- Knowledge Base Σ = <TBox, ABox>
- TBox
  - Conceptual or terminological knowledge
  - Intensional knowledge
  - General knowledge
  - Examples
    - Woman ≡ Person ∩ Female
    - Parent ≡ Person ∩ ∃ hasChild.Person ∩ ∀ hasChild.Person
    - Mother ≡ Female ∩ Parent
- ABox
  - Instance or assertional knowledge
  - Extensional knowledge
  - Knowledge in a situaion
  - Examples
    - Woman(Sazae)
    - hasChild(Sazae, Tara)






# Reasoning

- Subsumption
  - Concept satisfiablity: Σ |≠ C ≡ ⊥
  - Concept Subsumption: Σ |= C ⊆ D or Σ |= C ∩ ¬D≡ ⊥
  - Inconsistency:
  - Ex.)
    - Mother ⊆ Woman



# Structural Subsumption Algorithm

- Normalization (Conjunctive normal form)
  - Mother= Person ∩ Female ∩ ∀hasChild.Person
  - SMother= Person ∩ (∃hasChild ∩ ∀hasChild.Person) ∩ Female ∩ ∀ hasChild.Student
    = Female ∩ ∃ hasChild ∩ ∀ hasChild.(Person ∩ Student)
- Compare each term
  - C subsumes D if each term $C_i \in C$ satisfies:
    - If $C_i$ is atomic or ∃R, then there is $D_j$ with $D_j$=$C_i$
    - If $C_i$ is ∀R.C', then there is a $D_j$ with $D_j$= ∀R.D' and C' subsumes D'
- Liner complexity and sound
- Complete only for *FL*$^-$



# Tableau algorithms

- Check satisfiability of concept descriptions
  - Assume an instance ***b*** which satisfies all the descriptions
  - Then check whether this assumption turns out impossible
    - The assumption is wrong -> not satisfiable
- For NNF (Negation Nomarl Form)
  - Negation appears only just before concepts
- Completion rules
  - Adding constraints by interpreting terms
  - Derive contradiction

# Tableau algorithms

- Completion rules
  - $\cap$-rule:
    - Condition: S contains $(C\cap D)(x)$ and does not contains both $C(x)$ and $D(x)$
    - Action: $S'=S\cup\{C(x), D(x)\}$
  - $\cup$-rule:
    - Condition: S contains $(C\cup D)(x)$ and neither $C(x)$ nor $D(x)$
    - Action: $S'=S\cup\{C(x)\}$ or $S'=S\cup\{D(x)\}$
  - $\exists$-rule:
    - Condition: S contains $(\exists R.C)(x)$ and no individual z such that satisfy $C(z)$ and $R(x,z)$ in S
    - Action: $S'=S\cup\{C(y),R(x,y)\}$
  - $\forall$-rule:
    - Condition: S contains $(\forall R.C)(x)$ and $R(x,y)$, and does not contain $C(y)$
    - Action: $S'=S\cup\{C(y)\}$



# Sumsumption by Tableau algorithms

- Unfold Tbox *T* to *T'*
  - Remove defined concepts by applying their definition in ***T***
    - Pick up B such as $A\equiv B$ in ***T***
    - Replace A' such as $A'\equiv B'$ with B' in B recursively
- Remove defined concepts by applying their definition in $C\cap\neg D$
    - Pick up A such as $A\equiv B$ in $C\cap\neg D$
    - Replace A with B
- Transform $C\sqsubseteq D$ to $C\cap\neg D$
- Transform $C\cap\neg D$ into NNF
- Check $C\cap\neg D$ by Tableau algorithm

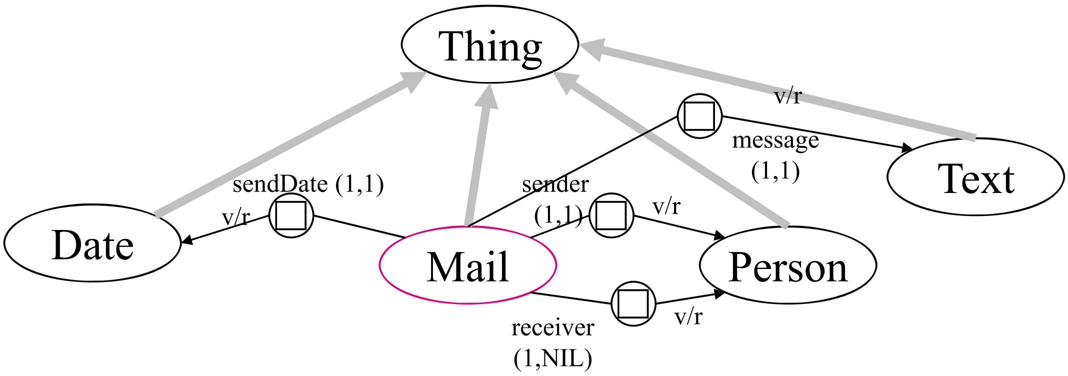
# Tableau algorithms: An exmple

- Dmom $\subseteq$ Mother ?
- T
  - Woman ≡ Person ∩ Female
  - Parent ≡ Person ∩ ∃ hasChild.Person ∩ ∀ hasChild.Person
  - Mother ≡ Female ∩ Parent
  - Dmom ≡ Woman ∩ ∃ hasChild.Woman ∩ ∀ hasChild.Woman
- T’
  - Woman ≡ Person ∩ Female
  - Parent ≡ Person ∩ ∃ hasChild.Person ∩ ∀ hasChild.Person
  - Mother ≡ Female ∩ Person ∩ ∃ hasChild.Person ∩ ∀ hasChild.Person
  - Dmom ≡ Person ∩ Female ∩ ∃ hasChild.(Person ∩ Female) ∩ ∀ hasChild. (Person ∩ Female)
- Transform C⊆D to C ∩¬D
  - Dmom ∩ ¬Mother
- Remove defined concepts
  - Person ∩ Female ∩ ∃ hasChild. (Person ∩ Female) ∩ ∀ hasChild. (Person ∩ Female) ∩ ¬(Female ∩ Person ∩ ∃ hasChild.Person ∩ ∀ hasChild.Person)
- NNF
  - Person ∩ Female ∩ ∃ hasChild. (Person ∩ Female) ∩ ∀ hasChild. (Person ∩ Female) ∩ (¬Female ∪ ¬Person ∪ ∀¬hasChild.Person ∪ ∃ ¬hasChild.Person)



# An example

- S0={x: Person ∩ Female ∩ ∃ hasChild. (Person ∩ Female) ∩ ∀ hasChild. (Person ∩ Female) ∩ (¬Female ∪ ¬Person ∪ ∀¬hasChild.Person ∪ ∃ ¬hasChild.Person)}
  - ∪-rule
  - S1=S0∪ {x: ¬Female}= ⊥
  - S1’’= S0∪ {x: ¬Person}= ⊥
  - S1’’’= S0∪ {x: ∀¬hasChild.Person }(1)
    - ∩-rule
  - S2=S1’’’∪ {x: Person ∩ Female ∩ ∃ hasChild. (Person ∩ Female), x:∀ hasChild. (Person ∩ Female)}
    - ∃-rule
  - S3=S2∪ {y: Person ∩Female, (x,y): hasChild}
    - ∩-rule
  - S4=S3∪{y:Person, y:Female}
    - ∀-rule (for (1))
  - S5= S4∪ {y: ¬Person}= ⊥
  - S1’’’’= S0∪ {x: ∃ ¬hasChild.Person }
    - ….




Mail $\sqsubseteq$ Thing $\sqcap$ $\forall$sendDate.Date $\sqcap$ $\forall$sender.Person $\sqcap$ $\forall$receiver.Persor