Applications of Description Logics
Terminological KR and Ontologies

DLs initially designed for terminological KR (and reasoning) is natural to use DLs to build and maintain ontologies.

Semantic Web

Semantic markup will be added to web resources. The aim is "machine-understandability". Markup will use ontologies to provide common terms of reference with clear semantics. This is a requirement for a web-based ontology language.

Well-defined semantics builds on existing Web standards (XML, RDF, RDFS). The resulting language (DAML+OIL) is based on a DL (SHIQ). DL reasoning can be used to support ontology design and maintenance, for example, to classify resources with respect to ontologies.
Application Areas

Terminological KR and Ontologies
Terminological KR and Ontologies

- DLs initially designed for terminological KR (and reasoning)
Terminological KR and Ontologies

- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies
Application Areas

- Terminological KR and Ontologies
  - DLs initially designed for terminological KR (and reasoning)
  - Natural to use DLs to build and maintain ontologies

- Semantic Web
Terminological KR and Ontologies
- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web
- **Semantic** markup will be added to web resources

Semantic markup will be added to web resources

Aim is "machine-understandability"
Markup will use Ontologies to provide common terms of reference with clear semantics
Requirement for web-based ontology language
Well-defined semantics
Builds on existing Web standards (XML, RDF, RDFS)
Resulting language (DAML+OIL) is based on a DL (SHIQ)
DL reasoning can be used to, e.g.,
- Support ontology design and maintenance
- Classify resources w.r.t. ontologies
Application Areas

Terminological KR and Ontologies
- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web
- **Semantic** markup will be added to web resources
  - Aim is “machine understandability”
Application Areas

- Terminological KR and Ontologies
  - DLs initially designed for terminological KR (and reasoning)
  - Natural to use DLs to build and maintain ontologies

- Semantic Web
  - **Semantic** markup will be added to web resources
    → Aim is “machine understandability”
  - Markup will use **Ontologies** to provide common terms of reference with clear semantics
Application Areas

Terminological KR and Ontologies
- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web
- **Semantic** markup will be added to web resources
  - Aim is “machine understandability”
- Markup will use **Ontologies** to provide common terms of reference with clear semantics
- Requirement for web based ontology language
Application Areas

Terminological KR and Ontologies
- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web
- **Semantic** markup will be added to web resources
  - Aim is “machine understandability”
- Markup will use **Ontologies** to provide common terms of reference with clear semantics
- Requirement for web based ontology language
  - Well defined semantics
Application Areas

Terminological KR and Ontologies
- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web
- **Semantic** markup will be added to web resources
  - Aim is “machine understandability”
- Markup will use **Ontologies** to provide common terms of reference with clear semantics
- Requirement for web based ontology language
  - Well defined semantics
  - Builds on existing Web standards (XML, RDF, RDFS)
Terminological KR and Ontologies
- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web
- **Semantic** markup will be added to web resources
  - Aim is “machine understandability”
- Markup will use **Ontologies** to provide common terms of reference with clear semantics
- Requirement for web based ontology language
  - Well defined semantics
  - Builds on existing Web standards (XML, RDF, RDFS)
- Resulting language (DAML+OIL) is **based on a DL** \((SHIQ)\)
Terminological KR and Ontologies

- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web

- **Semantic** markup will be added to web resources
  - Aim is “machine understandability”
- Markup will use **Ontologies** to provide common terms of reference with clear semantics
- Requirement for web based ontology language
  - Well defined semantics
  - Builds on existing Web standards (XML, RDF, RDFS)
- Resulting language (DAML+OIL) is **based on a DL** ($SHIQ$)
- DL **reasoning** can be used to, e.g.,
Application Areas I

Terminological KR and Ontologies
- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web
- **Semantic** markup will be added to web resources
  - Aim is "machine understandability"
- Markup will use **Ontologies** to provide common terms of reference with clear semantics
- Requirement for web based ontology language
  - Well defined semantics
  - Builds on existing Web standards (XML, RDF, RDFS)
- Resulting language (DAML+OIL) is **based on a DL** (*SHIQ*)
- DL **reasoning** can be used to, e.g.,
  - Support ontology design and maintenance
Terminological KR and Ontologies

- DLs initially designed for terminological KR (and reasoning)
- Natural to use DLs to build and maintain ontologies

Semantic Web

- **Semantic** markup will be added to web resources
  - Aim is “machine understandability”
- Markup will use **Ontologies** to provide common terms of reference with clear semantics
- Requirement for web based ontology language
  - Well defined semantics
  - Builds on existing Web standards (XML, RDF, RDFS)
- Resulting language (DAML+OIL) is **based on a DL** *(SHIQ)*
- DL **reasoning** can be used to, e.g.,
  - Support ontology design and maintenance
  - Classify resources w.r.t. ontologies
Application Areas II

- Classic system used to configure telecoms equipment
- Characteristics of components described in DL KB
- Reasoner checks validity (and price) of configurations
- LaSSIE system used DL KB for flexible software documentation and query answering
- Database applications
Application Areas II

- Configuration

- Classic system used to configure telecoms equipment
  - Characteristics of components described in DL KB
  - Reasoner checks validity (and price) of configurations

- LaSSIE system used DL KB for flexible software documentation and query answering

- Database applications
Application Areas

- Configuration
  - **Classic** system used to configure telecoms equipment
Application Areas II

Configuration

- **Classic** system used to configure telecoms equipment
- Characteristics of components described in DL KB
Application Areas II

- Configuration
  - **Classic** system used to configure telecoms equipment
  - Characteristics of components described in DL KB
  - Reasoner checks validity (and price) of configurations
Application Areas II

- Configuration
  - **Classic** system used to configure telecoms equipment
  - Characteristics of components described in DL KB
  - Reasoner checks validity (and price) of configurations

- Software information systems
Application Areas II

- **Configuration**
  - **Classic** system used to configure telecoms equipment
  - Characteristics of components described in DL KB
  - Reasoner checks validity (and price) of configurations

- **Software information systems**
  - LaSSIE system used DL KB for flexible software documentation and query answering
Application Areas

- **Configuration**
  - *Classic* system used to configure telecoms equipment
  - Characteristics of components described in DL KB
  - Reasoner checks validity (and price) of configurations

- **Software information systems**
  - LaSSIE system used DL KB for flexible software documentation and query answering

- **Database applications**
Application Areas

- **Configuration**
  - Classic system used to configure telecoms equipment
  - Characteristics of components described in DL KB
  - Reasoner checks validity (and price) of configurations

- **Software information systems**
  - LaSSIE system used DL KB for flexible software documentation and query answering

- **Database applications**

...
Database Schema and Query Reasoning

DL (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)

Satisfiability preserving mapping to SHIQ allows use of DL reasoners (e.g., FaCT, RACER)

DL Abox can also capture semantics of conjunctive queries
Can reason about query containment w.r.t. schema

DL reasoning can be used to support
Schema design, evolution and query optimisation
Source integration in heterogeneous databases/data warehouses
Conceptual modelling of multidimensional aggregation

E.g., I.COM Intelligent Conceptual Modelling tool (Enrico Franconi)
Uses FaCT system to provide reasoning support for EER
Database Schema and Query Reasoning

- $\mathcal{DLR}$ (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
Database Schema and Query Reasoning

- **DLR** (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to **SHIQ** allows use of DL reasoners (e.g., FaCT, RACER)

(C) 2001: Description Logics tutorial – p.4/9
**Database Schema and Query Reasoning**

- **DLR (n-ary DL)** can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to *SHIQ* allows use of DL reasoners (e.g., FaCT, RACER)
- DL Abox can also capture semantics of conjunctive queries

*E.g., I.COM Intelligent Conceptual Modelling tool (Enrico Franconi)*

*Uses FaCT system to provide reasoning support for EER*
Database Schema and Query Reasoning

- DLR (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to SHIQ allows use of DL reasoners (e.g., FaCT, RACER)
- DL Abox can also capture semantics of conjunctive queries
  - Can reason about query containment w.r.t. schema
Database Schema and Query Reasoning

- DLR (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to SHIQ allows use of DL reasoners (e.g., FaCT, RACER)
- DL Abox can also capture semantics of conjunctive queries
  - Can reason about query containment w.r.t. schema
- DL reasoning can be used to support schema design, evolution and query optimisation, source integration in heterogeneous databases/data warehouses, and conceptual modelling of multidimensional aggregation

E.g., I.COM (Intelligent Conceptual Modelling tool by Enrico Franconi) uses FaCT system to provide reasoning support for EER
Database Schema and Query Reasoning

- DLR (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to SHIQ allows use of DL reasoners (e.g., FaCT, RACER)
- DL Abox can also capture semantics of conjunctive queries
  - Can reason about query containment w.r.t. schema
- DL reasoning can be used to support
  - Schema design, evolution and query optimisation
**Database Schema and Query Reasoning**

- **DLR (n-ary DL)** can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to **SHIQ** allows use of DL reasoners (e.g., FaCT, RACER)
- **DL Abox** can also capture semantics of conjunctive queries
  - Can reason about query containment w.r.t. schema
- **DL reasoning** can be used to support
  - Schema design, evolution and query optimisation
  - Source integration in heterogeneous databases/data warehouses
Database Schema and Query Reasoning

- DLR (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to SHIQ allows use of DL reasoners (e.g., FaCT, RACER)
- DL Abox can also capture semantics of conjunctive queries
  - Can reason about query containment w.r.t. schema
- DL reasoning can be used to support
  - Schema design, evolution and query optimisation
  - Source integration in heterogeneous databases/data warehouses
  - Conceptual modelling of multidimensional aggregation
**Database Schema and Query Reasoning**

- **DLR** (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to **SHIQ** allows use of DL reasoners (e.g., FaCT, RACER)
- DL Abox can also capture semantics of conjunctive queries
  - Can reason about query containment w.r.t. schema
- DL reasoning can be used to support
  - Schema design, evolution and query optimisation
  - Source integration in heterogeneous databases/data warehouses
  - Conceptual modelling of multidimensional aggregation
- E.g., **I.COM** Intelligent Conceptual Modelling tool (Enrico Franconi)
Database Schema and Query Reasoning

- *DLR* (n-ary DL) can capture semantics of many conceptual modelling methodologies (e.g., EER)
- Satisfiability preserving mapping to *SHIQ* allows use of DL reasoners (e.g., FaCT, RACER)
- DL Abox can also capture semantics of conjunctive queries
  - Can reason about query containment w.r.t. schema
- DL reasoning can be used to support
  - Schema design, evolution and query optimisation
  - Source integration in heterogeneous databases/data warehouses
  - Conceptual modelling of multidimensional aggregation
- E.g., **I.COM** Intelligent Conceptual Modelling tool (Enrico Franconi)
  - Uses FaCT system to provide reasoning support for EER
I.COM Demo
Initial motivation for work on FaCT system was Galen project. General requirement for medical terminologies is that static lists/taxonomies are difficult to build and maintain. They need to be very large and highly interconnected, inevitably containing many errors and omissions. Galen project aims to replace static hierarchy with DL to describe concepts (e.g., spiral fracture of left femur). Use DL classifier to build taxonomy. Needed expressive DL and efficient reasoning. Descriptions use transitive/inverse roles, GCIs etc. Very large KBs (tens of thousands of concepts). Even prototype KB is very large (3,000 concepts). Existing (incomplete) classifier took 24 hours to classify KB. FaCT system (sound and complete) takes 60 seconds.
Initial motivation for work on FaCT system was Galen project
Initial motivation for work on FaCT system was Galen project
☞ General requirement for medical terminologies
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
  - Need to be very large and highly interconnected
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
  - Need to be very large and highly interconnected
  - Inevitably contain many errors and omissions
Initial motivation for work on **FaCT** system was **Galen** project

☞ General requirement for medical terminologies

☞ Static lists/taxonomies difficult to build and maintain
  - Need to be very **large** and highly interconnected
  - Inevitably contain many **errors** and **omissions**

☞ Galen project aims to replace static hierarchy with DL
Initial motivation for work on FaCT system was Galen project

☞ General requirement for medical terminologies
☞ Static lists/taxonomies difficult to build and maintain
  ● Need to be very large and highly interconnected
  ● Inevitably contain many errors and omissions
☞ Galen project aims to replace static hierarchy with DL
  ● Describe concepts (e.g., spiral fracture of left femur)
Terminological KR and Ontologies

Initial motivation for work on FaCT system was Galen project

☞ General requirement for medical terminologies
☞ Static lists/taxonomies difficult to build and maintain
  ● Need to be very large and highly interconnected
  ● Inevitably contain many errors and omissions
☞ Galen project aims to replace static hierarchy with DL
  ● Describe concepts (e.g., spiral fracture of left femur)
  ● Use DL classifier to build taxonomy
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
  - Need to be very large and highly interconnected
  - Inevitably contain many errors and omissions
- Galen project aims to replace static hierarchy with DL
  - Describe concepts (e.g., spiral fracture of left femur)
  - Use DL classifier to build taxonomy
- Needed expressive DL and efficient reasoning
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
  - Need to be very large and highly interconnected
  - Inevitably contain many errors and omissions
- Galen project aims to replace static hierarchy with DL
  - Describe concepts (e.g., spiral fracture of left femur)
  - Use DL classifier to build taxonomy
- Needed expressive DL and efficient reasoning
  - Descriptions use transitive/inverse roles, GCIs etc.
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
  - Need to be very large and highly interconnected
  - Inevitably contain many errors and omissions
- Galen project aims to replace static hierarchy with DL
  - Describe concepts (e.g., spiral fracture of left femur)
  - Use DL classifier to build taxonomy
- Needed expressive DL and efficient reasoning
  - Descriptions use transitive/inverse roles, GCIs etc.
  - Very large KBs (tens of thousands of concepts)
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
  - Need to be very large and highly interconnected
  - Inevitably contain many errors and omissions
- Galen project aims to replace static hierarchy with DL
  - Describe concepts (e.g., spiral fracture of left femur)
  - Use DL classifier to build taxonomy
- Needed expressive DL and efficient reasoning
  - Descriptions use transitive/inverse roles, GCIs etc.
  - Very large KBs (tens of thousands of concepts)
    ➔ Even prototype KB is very large (≈3,000 concepts)
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
  - Need to be very **large** and highly interconnected
  - Inevitably contain many **errors** and **omissions**
- Galen project aims to replace static hierarchy with DL
  - **Describe** concepts (e.g., spiral fracture of left femur)
  - Use DL classifier to **build taxonomy**
- Needed expressive DL **and** efficient reasoning
  - Descriptions use transitive/inverse roles, GCIs etc.
  - Very large KBs (tens of thousands of concepts)
    - Even prototype KB is very large ($\approx 3,000$ concepts)
    - Existing (incomplete) classifier took $\approx 24$ **hours** to classify KB
Initial motivation for work on FaCT system was Galen project

- General requirement for medical terminologies
- Static lists/taxonomies difficult to build and maintain
  - Need to be very large and highly interconnected
  - Inevitably contain many errors and omissions
- Galen project aims to replace static hierarchy with DL
  - Describe concepts (e.g., spiral fracture of left femur)
  - Use DL classifier to build taxonomy
- Needed expressive DL and efficient reasoning
  - Descriptions use transitive/inverse roles, GCIs etc.
  - Very large KBs (tens of thousands of concepts)
    - Even prototype KB is very large (≈3,000 concepts)
    - Existing (incomplete) classifier took ≈24 hours to classify KB
    - FaCT system (sound and complete) takes ≈60 seconds
DL reasoner can be used to support design and maintenance.

Example is OilEd ontology editor (for DAML+OIL).

Frame based interface (like Protegé, OntoEdit, etc.) extended to clarify semantics and capture whole DAML+OIL language.

- Slots explicitly existential or value restrictions
- Boolean connectives and nesting
- Properties for slot relations (transitive, functional etc.)
- General axioms

Reasoning support for OilEd provided by FaCT system.

Frame representation translated into SHIQ.

Communicates with FaCT via CORBA interface.

Indicates inconsistencies and implicit subsumptions.

Can make implicit subsumptions explicit in KB.
DL reasoner can be used to support design and maintenance.
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
DL reasoner can be used to support design and maintenance.

Example is OilEd ontology editor (for DAML+OIL)
- Frame based interface (like Protegé, OntoEdit, etc.)
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
  - Frame based interface (like Protegé, OntoEdit, etc.)
  - Extended to clarify semantics and capture whole DAML+OIL language
DL reasoner can be used to support design and maintenance.

Example is OilEd ontology editor (for DAML+OIL)
- Frame based interface (like Protegé, OntoEdit, etc.)
- Extended to clarify semantics and capture whole DAML+OIL language
  - Slots explicitly existential or value restrictions
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
  - Frame based interface (like Protegé, OntoEdit, etc.)
  - Extended to clarify semantics and capture whole DAML+OIL language
    - Slots explicitly existential or value restrictions
    - Boolean connectives and nesting

Reasoning support for OilEd provided by FaCT system
Frame representation translated into SHIQ
Communicates with FaCT via CORBA interface
Indicates inconsistencies and implicit subsumptions
Can make implicit subsumptions explicit in KB
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
  - Frame based interface (like Protegé, OntoEdit, etc.)
  - Extended to clarify semantics and capture whole DAML+OIL language
    - Slots explicitly existential or value restrictions
    - Boolean connectives and nesting
    - Properties for slot relations (transitive, functional etc.)
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
  - Frame based interface (like Protegé, OntoEdit, etc.)
  - Extended to clarify semantics and capture whole DAML+OIL language
    - Slots explicitly existential or value restrictions
    - Boolean connectives and nesting
    - Properties for slot relations (transitive, functional etc.)
    - General axioms
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
  - Frame based interface (like Protegé, OntoEdit, etc.)
  - Extended to clarify semantics and capture whole DAML+OIL language
    - Slots explicitly existential or value restrictions
    - Boolean connectives and nesting
    - Properties for slot relations (transitive, functional etc.)
    - General axioms
- Reasoning support for OilEd provided by FaCT system
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
  - Frame based interface (like Protegé, OntoEdit, etc.)
  - Extended to clarify semantics and capture whole DAML+OIL language
    - Slots explicitly existential or value restrictions
    - Boolean connectives and nesting
    - Properties for slot relations (transitive, functional etc.)
    - General axioms
- Reasoning support for OilEd provided by FaCT system
  - Frame representation translated into $\mathcal{SHIQ}$
DL reasoner can be used to support design and maintenance.

Example is OilEd ontology editor (for DAML+OIL)
- Frame based interface (like Protegé, OntoEdit, etc.)
- Extended to clarify semantics and capture whole DAML+OIL language
  - Slots explicitly existential or value restrictions
  - Boolean connectives and nesting
  - Properties for slot relations (transitive, functional etc.)
  - General axioms

Reasoning support for OilEd provided by FaCT system
- Frame representation translated into $SHIQ$
- Communicates with FaCT via CORBA interface
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
  - Frame based interface (like Protegé, OntoEdit, etc.)
  - Extended to clarify semantics and capture whole DAML+OIL language
    - Slots explicitly existential or value restrictions
    - Boolean connectives and nesting
    - Properties for slot relations (transitive, functional etc.)
    - General axioms
- Reasoning support for OilEd provided by FaCT system
  - Frame representation translated into $SHIQ$
  - Communicates with FaCT via CORBA interface
  - Indicates inconsistencies and implicit subsumptions
Reasoning Support for Ontology Design

- DL reasoner can be used to support design and maintenance
- Example is OilEd ontology editor (for DAML+OIL)
  - Frame based interface (like Protegé, OntoEdit, etc.)
  - Extended to clarify semantics and capture whole DAML+OIL language
    - Slots explicitly existentially or value restrictions
    - Boolean connectives and nesting
    - Properties for slot relations (transitive, functional etc.)
    - General axioms
- Reasoning support for OilEd provided by FaCT system
  - Frame representation translated into $SHIQ$
  - Communicates with FaCT via CORBA interface
  - Indicates inconsistencies and implicit subsumptions
  - Can make implicit subsumptions explicit in KB
E.g., DAML+OIL medical terminology ontology

- Transitive roles capture transitive partonomy, causality, etc.

  - Smoking causes Cancer + Cancer causes Death

  - GCIs represent additional non-definitional knowledge

    - Stomach-Ulcer = Ulcer hasLocation Stomach + Stomach-Ulcer hasLocation Lining-Of-Stomach

- Inverse roles capture e.g. causes/causedBy relationship

  - Death causedBy Smoking + Smoking CauseOfPrematureDeath

- Cardinality restrictions add consistency constraints

  - BloodPressure hasValue (High t Low) u 6 hasValue + High v : Low

HighLowBloodPressure v ????
DAML+OIL Medical Terminology Examples

E.g., DAML+OIL medical terminology ontology

- Transitive roles capture transitive partonomy, causality, etc.
E.g., DAML+OIL medical terminology ontology

Transitive roles capture transitive partonomy, causality, etc.

Smoking $\sqsubseteq \exists$ causes.Cancer plus Cancer $\sqsubseteq \exists$ causes.Death

$\Rightarrow$ Cancer $\sqsubseteq$ FatalThing
DAML+OIL Medical Terminology Examples

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture transitive partonomy, causality, etc.
   Smoking $\sqsubseteq \exists$causes.Cancer plus Cancer $\sqsubseteq \exists$causes.Death
   $\Rightarrow$ Cancer $\sqsubseteq$ FatalThing

☞ GCIs represent additional non-definitional knowledge
DAML+OIL Medical Terminology Examples

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture transitive partonomy, causality, etc.

\[
\text{Smoking} \sqsubseteq \exists \text{causes}.\text{Cancer} \quad \text{plus} \quad \text{Cancer} \sqsubseteq \exists \text{causes}.\text{Death} \\
\Rightarrow \text{Cancer} \sqsubseteq \text{FatalThing}
\]

☞ GCIs represent additional non-definitional knowledge

\[
\text{Stomach-Ulcer} \models \text{Ulcer} \sqcap \exists \text{hasLocation}.\text{Stomach} \quad \text{plus} \\
\text{Stomach-Ulcer} \sqsubseteq \exists \text{hasLocation}.\text{Lining-Of-Stomach} \\
\Rightarrow \text{Ulcer} \sqcap \exists \text{hasLocation}.\text{Stomach} \sqsubseteq \text{OrganLiningLesion}
\]
E.g., DAML+OIL medical terminology ontology

- Transitive roles capture transitive partonomy, causality, etc.
  
  Smoking ⊑ ∃causes.Cancer plus Cancer ⊑ ∃causes.Death
  ⇒ Cancer ⊑ FatalThing

- GCIs represent additional non-definitional knowledge
  
  Stomach-Ulcer ≜ Ulcer ⊑ ∃hasLocation.Stomach plus
  Stomach-Ulcer ⊑ ∃hasLocation.Lining-Of-Stomach
  ⇒ Ulcer ⊑ ∃hasLocation.Stomach ⊑ OrganLiningLesion

- Inverse roles capture e.g. causes/causedBy relationship
DAML+OIL Medical Terminology Examples

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture transitive partonomy, causality, etc.

\[
\text{Smoking} \sqsubseteq \exists \text{causes}.\text{Cancer} \quad \text{plus} \quad \text{Cancer} \sqsubseteq \exists \text{causes}.\text{Death}
\Rightarrow \text{Cancer} \sqsubseteq \text{FatalThing}
\]

☞ GCIs represent additional non-definitional knowledge

\[
\text{Stomach-Ulcer} \sqsubseteq \text{Ulcer} \sqcap \exists \text{hasLocation}.\text{Stomach} \quad \text{plus}
\text{Stomach-Ulcer} \sqsubseteq \exists \text{hasLocation}.\text{Lining-Of-Stomach}
\Rightarrow \text{Ulcer} \sqcap \exists \text{hasLocation}.\text{Stomach} \sqsubseteq \text{OrganLiningLesion}
\]

☞ Inverse roles capture e.g. causes/causedBy relationship

\[
\text{Death} \sqcap \exists \text{causedBy}.\text{Smoking} \sqsubseteq \text{PrematureDeath}
\Rightarrow \text{Smoking} \sqsubseteq \text{CauseOfPrematureDeath}
\]
E.g., DAML+OIL medical terminology ontology

Transitive roles capture transitive partonomy, causality, etc.

Smoking ⊑ ∃causes.Cancer plus Cancer ⊑ ∃causes.Death
⇒ Cancer ⊑ FatalThing

GCIs represent additional non-definitional knowledge

Stomach-Ulcer ≡ Ulcer ⊑ ∃hasLocation.Stomach plus
Stomach-Ulcer ⊑ ∃hasLocation.Lining-Of-Stomach
⇒ Ulcer ⊑ ∃hasLocation.Stomach ⊑ OrganLiningLesion

Inverse roles capture e.g. causes/causedBy relationship

Death ⊑ ∃causedBy.Smoking ⊑ PrematureDeath
⇒ Smoking ⊑ CauseOfPrematureDeath

Cardinality restrictions add consistency constraints
DAML+OIL Medical Terminology Examples

E.g., DAML+OIL medical terminology ontology

Transitive roles capture transitive partonomy, causality, etc.

\[ \text{Smoking} \sqsubseteq \exists \text{cause}.\text{Cancer} \text{ plus } \exists \text{cause}.\text{Death} \Rightarrow \text{Cancer} \sqsubseteq \exists \text{FatalThing} \]

\[ \text{GCIs represent additional non-definitional knowledge} \]

\[ \text{Stomach-Ulcer} \doteq \text{Ulcer} \sqcap \exists \text{hasLocation}.\text{Stomach} \text{ plus } \text{Stomach-Ulcer} \sqsubseteq \exists \text{hasLocation}.\text{Lining-Of-Stomach} \Rightarrow \text{Ulcer} \sqcap \exists \text{hasLocation}.\text{Stomach} \sqsubseteq \text{OrganLiningLesion} \]

Inverse roles capture e.g. causes/causedBy relationship

\[ \text{Death} \sqcap \exists \text{causedBy}.\text{Smoking} \sqsubseteq \text{PrematureDeath} \Rightarrow \text{Smoking} \sqsubseteq \text{CauseOfPrematureDeath} \]

\[ \text{Cardinality restrictions add consistency constraints} \]

\[ \text{BloodPressure} \sqsubseteq \exists \text{hasValue}.(\text{High} \sqcup \text{Low}) \sqsubseteq \text{1hasValue} \text{ plus } \text{High} \sqsubseteq \neg \text{Low} \Rightarrow \text{HighLowBloodPressure} \sqsubseteq \bot \]