Reasoning with DAML+OIL:
What can it do for YOU?

Ian Horrocks
horrocks@cs.man.ac.uk

University of Manchester
Manchester, UK
DAML+OIL Language Overview

DAML+OIL is an ontology language
DAML+OIL Language Overview

DAML+OIL is an **ontology** language

☞ Describes **structure** of the domain (i.e., a schema)
  ● RDF used to describe specific **instance** of domain (data)
DAML+OIL Language Overview

DAML+OIL is an **ontology** language

☞ Describes **structure** of the domain (i.e., a schema)
  - RDF used to describe specific **instance** of domain (data)

☞ Structure described in terms of **classes** and **properties**
DAML+OIL Language Overview

DAML+OIL is an **ontology** language

☞ Describes **structure** of the domain (i.e., a schema)
  - RDF used to describe specific **instance** of domain (data)

☞ Structure described in terms of **classes** and **properties**

☞ Ontology consists of set of **axioms**
  - E.g., asserting class subsumption/equivalence
DAML+OIL is an **ontology** language

☞ Describes **structure** of the domain (i.e., a schema)
  - RDF used to describe specific **instance** of domain (data)

☞ Structure described in terms of **classes** and **properties**

☞ Ontology consists of set of **axioms**
  - E.g., asserting class subsumption/equivalence

☞ Classes can be names or **expressions**
  - Various **constructors** provided for building class expressions
DAML+OIL Language Overview

DAML+OIL is an **ontology** language

☞ Describes **structure** of the domain (i.e., a schema)
  * RDF used to describe specific **instance** of domain (data)

☞ Structure described in terms of **classes** and **properties**

☞ Ontology consists of set of **axioms**
  * E.g., asserting class subsumption/equivalence

☞ Classes can be names or **expressions**
  * Various **constructors** provided for building class expressions

☞ **Expressive power** determined by
  * Kinds of axiom supported
  * Kinds of class (and property) constructor supported
### DAML+OIL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Abbreviation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \land \ldots \land C_n$</td>
<td>Human $\land$ Male</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \lor \ldots \lor C_n$</td>
<td>Doctor $\lor$ Lawyer</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1 \ldots x_n}$</td>
<td>${john, mary}$</td>
</tr>
<tr>
<td>toClass</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Dr.</td>
</tr>
<tr>
<td>hasClass</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lw.</td>
</tr>
<tr>
<td>hasValue</td>
<td>$\exists P.{x}$</td>
<td>$\exists$ citizenOf.USA</td>
</tr>
<tr>
<td>minCardinalityQ</td>
<td>$\geq n \ P.C$</td>
<td>$\geq 2$ hasChild.Lw.</td>
</tr>
<tr>
<td>maxCardinalityQ</td>
<td>$\leq n \ P.C$</td>
<td>$\leq 1$ hasChild.Male</td>
</tr>
<tr>
<td>cardinalityQ</td>
<td>$= n \ P.C$</td>
<td>$= 1$ hasParent.Female</td>
</tr>
</tbody>
</table>
## DAML+OIL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Abbreviation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \land \ldots \land C_n$</td>
<td>Human $\land$ Male</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \lor \ldots \lor C_n$</td>
<td>Doctor $\lor$ Lawyer</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1 \ldots x_n}$</td>
<td>{john, mary}</td>
</tr>
<tr>
<td>toClass</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
</tr>
<tr>
<td>hasClass</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
</tr>
<tr>
<td>hasValue</td>
<td>$\exists P.{x}$</td>
<td>$\exists$ citizenOf.{USA}</td>
</tr>
<tr>
<td>minCardinalityQ</td>
<td>$\geq n\ P.C$</td>
<td>$\geq 2$ hasChild.Lawyer</td>
</tr>
<tr>
<td>maxCardinalityQ</td>
<td>$\leq n\ P.C$</td>
<td>$\leq 1$ hasChild.Male</td>
</tr>
<tr>
<td>cardinalityQ</td>
<td>$= n\ P.C$</td>
<td>$= 1$ hasParent.Female</td>
</tr>
</tbody>
</table>

☞ Arbitrarily complex **nesting** of constructors

- E.g., $\forall$ hasChild. (Doctor $\lor$ $\exists$ hasChild.Doctor)
### DAML+OIL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Abbreviation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \land \ldots \land C_n$</td>
<td>Human $\land$ Male</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \lor \ldots \lor C_n$</td>
<td>Doctor $\lor$ Lawyer</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1 \ldots x_n}$</td>
<td>${\text{john, mary}}$</td>
</tr>
<tr>
<td>toClass</td>
<td>$\forall P.C$</td>
<td>$\forall \text{hasChild.Doctor}$</td>
</tr>
<tr>
<td>hasClass</td>
<td>$\exists P.C$</td>
<td>$\exists \text{hasChild.Lawyer}$</td>
</tr>
<tr>
<td>hasValue</td>
<td>$\exists P.{x}$</td>
<td>$\exists \text{citizenOf.{USA}}$</td>
</tr>
<tr>
<td>minCardinalityQ</td>
<td>$\geq n P.C$</td>
<td>$\geq 2 \text{hasChild.Lawyer}$</td>
</tr>
<tr>
<td>maxCardinalityQ</td>
<td>$\leq n P.C$</td>
<td>$\leq 1 \text{hasChild.Male}$</td>
</tr>
<tr>
<td>cardinalityQ</td>
<td>$= n P.C$</td>
<td>$= 1 \text{hasParent.Female}$</td>
</tr>
</tbody>
</table>

- Arbitrarily complex **nesting** of constructors
  - E.g., $\forall \text{hasChild.}(\text{Doctor} \lor \exists \text{hasChild.Doctor})$

- **XMLS datatypess** as well as classes
### DAML+OIL Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Abbreviation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\land$ Biped</td>
</tr>
<tr>
<td>sameClassAs</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\land$ Male</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>samePropertyAs</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>$x_1 \equiv x_2$</td>
<td>President_Bush $\equiv$ G_W_Bush</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \sqsubseteq \neg C_2$</td>
<td>Male $\sqsubseteq \neg$ Female</td>
</tr>
<tr>
<td>differentIndividualFrom</td>
<td>${x_1} \sqsubseteq \neg {x_2}$</td>
<td>${john} \sqsubseteq \neg {peter}$</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \equiv P_2$</td>
<td>hasChild $\equiv$ hasParent$^-$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+$ $\sqsubseteq$ ancestor</td>
</tr>
<tr>
<td>uniqueProperty</td>
<td>Thing $\sqsubseteq \leq_1 P$</td>
<td>Thing $\sqsubseteq \leq_1$ hasMother</td>
</tr>
<tr>
<td>UnambiguousProperty</td>
<td>Thing $\sqsubseteq \leq_1 P^-$</td>
<td>Thing $\sqsubseteq \leq_1$ isMotherOf$^-$</td>
</tr>
</tbody>
</table>
### DAML+OIL Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Abbreviation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\land$ Biped</td>
</tr>
<tr>
<td>sameClassAs</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\land$ Male</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>samePropertyAs</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>$x_1 \equiv x_2$</td>
<td>President_Bush $\equiv$ G_W_Bush</td>
</tr>
<tr>
<td>disjointWith</td>
<td>$C_1 \sqsubseteq \neg C_2$</td>
<td>Male $\sqsubseteq \neg$ Female</td>
</tr>
<tr>
<td>differentIndividualFrom</td>
<td>${x_1} \sqsubseteq \neg {x_2}$</td>
<td>${john} \sqsubseteq \neg {peter}$</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P_1 \equiv P_2^\perp$</td>
<td>hasChild $\equiv$ hasParent $\perp$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+ \sqsubseteq$ ancestor</td>
</tr>
<tr>
<td>uniqueProperty</td>
<td>Thing $\sqsubseteq \leq_1 P$</td>
<td>Thing $\sqsubseteq \leq_1$hasMother</td>
</tr>
<tr>
<td>UnambiguousProperty</td>
<td>Thing $\sqsubseteq \leq_1 P^\perp$</td>
<td>Thing $\sqsubseteq \leq_1 isMotherOf$</td>
</tr>
</tbody>
</table>

☞ Axioms (mostly) **reducible to subClass/PropertyOf**
Decidable Reasoning

Set of operators/axioms restricted so that reasoning is **decidable**
Decidable Reasoning

Set of operators/axioms restricted so that reasoning is **decidable**

☞ Significant point on tractability -v- expressiveness scale
Decidable Reasoning

Set of operators/axioms restricted so that reasoning is **decidable**

☞ Significant point on tractability -v- expressiveness scale

☞ Consistent with Semantic Web’s **layered architecture**
  - XML provides syntax transport layer
  - RDF provides basic ontological primitives
  - DAML+OIL provides (decidable) logical layer
  - Further layers (e.g., **rules**) will extend DAML+OIL
Decidable Reasoning

Set of operators/axioms restricted so that reasoning is **decidable**

- Significant point on tractability -v- expressiveness scale
- Consistent with Semantic Web’s **layered architecture**
  - XML provides syntax transport layer
  - RDF provides basic ontological primitives
  - DAML+OIL provides (decidable) logical layer
  - Further layers (e.g., **rules**) will extend DAML+OIL

- Facilitates provision of **reasoning services**
  - Known algorithms
  - Implemented systems
  - Evidence of **empirical tractability**
Why Reasoning Services?

Reasoning is important for:

- **Ontology design**: Check class consistency and (unexpected) implied relationships. Particularly important with large ontologies/multiple authors.

- **Ontology integration**: Assert inter-ontology relationships. Reasoner computes integrated class hierarchy/consistency.

- **Ontology deployment**: Determine if set of facts are consistent w.r.t. ontology. Determine if individuals are instances of ontology classes. No point in having semantics unless exploited by "agents".

"The Semantic Web needs a logic on top" (Henry Thompson)
Why Reasoning Services?

Reasoning is important for:

- Ontology **design**
  - Check class consistency and (unexpected) implied relationships
  - Particularly important with large ontologies/multiple authors

"The Semantic Web needs a logic on top"

(Henry Thompson)

DAML PI meeting, Nashua, July 2001 – p.6/9
Why Reasoning Services?

Reasoning is important for:

☞ Ontology design
  - Check class consistency and (unexpected) implied relationships
  - Particularly important with large ontologies/multiple authors

☞ Ontology integration
  - Assert inter-ontology relationships
  - Reasoner computes integrated class hierarchy/consistency
Why Reasoning Services?

Reasoning is important for:

☞ Ontology **design**
  - Check class consistency and (unexpected) implied relationships
  - Particularly important with large ontologies/multiple authors

☞ Ontology **integration**
  - Assert inter-ontology relationships
  - Reasoner computes integrated class hierarchy/consistency

☞ Ontology **deployment**
  - Determine if set of facts are consistent w.r.t. ontology
  - Determine if individuals are instances of ontology classes
  - No point in having semantics unless exploited by “agents”
Why Reasoning Services?

Reasoning is important for:

☞ **Ontology design**
  - Check class consistency and (unexpected) implied relationships
  - Particularly important with large ontologies/multiple authors

☞ **Ontology integration**
  - Assert inter-ontology relationships
  - Reasoner computes integrated class hierarchy/consistency

☞ **Ontology deployment**
  - Determine if set of facts are consistent w.r.t. ontology
  - Determine if individuals are instances of ontology classes
  - No point in having semantics unless exploited by “agents”

“The Semantic Web needs a logic on top” (Henry Thompson)
OilEd

OilEd is a DAML+OIL **ontology editor** with reasoning support.
OilEd is a DAML+OIL **ontology editor** with reasoning support

- Frame based interface (inspired by Protegé)
OilEd is a DAML+OIL **ontology editor** with reasoning support

- Frame based interface (inspired by Protegé)
- Extended to clarify semantics and capture whole language
  - Explicit $\exists$ (hasClass) or $\forall$ (toClass) restrictions
  - Boolean connectives ($\land$, $\lor$, $\neg$) and nesting
  - Transitive and unique (functional) properties

Reasoning support provided by FaCT system

Ontology translated into SHIQ DL

Communicates with FaCT via CORBA interface

Indicates inconsistencies and implicit subsumptions

Can add axioms to make implicit subsumptions explicit
OilEd

OilEd is a DAML+OIL ontology editor with reasoning support

☞ Frame based interface (inspired by Protegé)
☞ Extended to clarify semantics and capture whole language
  ● Explicit ∃ (hasClass) or ∀ (toClass) restrictions
  ● Boolean connectives (∧, ∨, ¬) and nesting
  ● Transitive and unique (functional) properties
☞ Reasoning support provided by FaCT system
  ● Ontology translated into $\mathcal{SHIQ}$ DL
  ● Communicates with FaCT via CORBA interface
  ● Indicates inconsistencies and implicit subsumptions
  ● Can add axioms to make implicit subsumptions explicit
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology

- Transitive roles capture partonomy, causality, etc.
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture partonomy, causality, etc.

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

$\Rightarrow$ Smoking $\sqsubseteq \exists$causes.Death
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology

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

☞ Multiple equality/inclusion axioms
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture partonomy, causality, etc.

Smoking $\subseteq \exists \text{causes}.\text{Cancer}$ plus $\text{Cancer} \subseteq \exists \text{causes}.\text{Death}$

$\Rightarrow$ Smoking $\subseteq \exists \text{causes}.\text{Death}$

☞ Multiple equality/inclusion axioms

Stomach-Ulcer $\cong \text{Ulcer} \land \exists \text{hasLocation}.\text{Stomach}$ plus

Stomach-Ulcer $\subseteq \exists \text{hasLocation}.\text{Lining-Of-Stomach}$

$\Rightarrow$ Ulcer $\land \exists \text{hasLocation}.\text{Stomach} \subseteq \text{OrganLiningLesion}$
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture partonomy, causality, etc.

   Smoking ⊆ ∃causes.Cancer plus Cancer ⊆ ∃causes.Death
   ⇒ Smoking ⊆ ∃causes.Death

☞ Multiple equality/inclusion axioms

   Stomach-Ulcer = Ulcer ∧ ∃hasLocation.Stomach plus
   Stomach-Ulcer ⊆ ∃hasLocation.Lining-Of-Stomach
   ⇒ Ulcer ∧ ∃hasLocation.Stomach ⊆ OrganLiningLesion

☞ Inverse roles capture e.g. causes/causedBy relationship
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture partonomy, causality, etc.

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

☞ Multiple equality/inclusion axioms

Stomach-Ulcer $\doteq$ Ulcer $\land \exists$hasLocation.Stomach plus
Stomach-Ulcer $\sqsubseteq \exists$hasLocation.Lining-Of-Stomach
⇒ Ulcer $\land \exists$hasLocation.Stomach $\sqsubseteq$ OrganLiningLesion

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

Death $\land \exists$causedBy.Smoking $\sqsubseteq$ PrematureDeath
⇒ Smoking $\sqsubseteq \exists$causes.PrematureDeath
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology

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

☞ Multiple equality/inclusion axioms
   Stomach-Ulcer $\triangleq$ Ulcer $\land \exists$hasLocation.Stomach plus
   Stomach-Ulcer $\sqsubseteq \exists$hasLocation.Lining-Of-Stomach
   $\Rightarrow$ Ulcer $\land \exists$hasLocation.Stomach $\sqsubseteq$ OrganLiningLesion

☞ Inverse roles capture e.g. causes/causedBy relationship
   Death $\land \exists$causedBy.Smoking $\sqsubseteq$ PrematureDeath
   $\Rightarrow$ Smoking $\sqsubseteq \exists$causes.PrematureDeath

☞ Cardinality restrictions add consistency constraints
Reasoning Examples — what you CAN do

E.g., DAML+OIL medical terminology ontology

☞ Transitive roles capture partonomy, causality, etc.

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

☞ Multiple equality/inclusion axioms

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

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

\[
\text{Death} \land \exists \text{causedBy}.\text{Smoking} \sqsubseteq \text{PrematureDeath}
\]
\[
\Rightarrow \text{Smoking} \sqsubseteq \exists \text{causes}.\text{PrematureDeath}
\]

☞ Cardinality restrictions add consistency constraints

\[
\text{BloodPressure} \sqsubseteq \exists \text{hasValue}.(\text{High} \lor \text{Low}) \land \leq_1 \text{hasValue} \quad \text{plus}
\]
\[
\text{High} \sqsubseteq \neg \text{Low} \Rightarrow \text{HighLowBloodPressure} \sqsubseteq \bot
\]
Reasoning Examples — what you CAN’T do

Where to begin!

Robust decidability largely due to tree model property.
For any consistent class there exists a tree (like) model.

No property constructors, e.g.:
- parent
- brother
- uncle
- ancestor:

No variables, e.g.:
- Ulcer

Only have unary and binary predicates; can't express (directly)

Language extensions may remove some of above limitations.

But there is no such thing as a free lunch.
Reasoning Examples — what you CAN’T do

Where to begin!

☞ Robust decidability largely due to tree model property
  ● For any consistent class there exists a tree (like) model

Language extensions may remove some of above limitations

☞ But there is no such thing as a free lunch
Reasoning Examples — what you CAN’T do

Where to begin!

☞ Robust decidability largely due to tree model property
  ● For any consistent class there exists a tree (like) model

☞ No property constructors, e.g.:
  ● parent ♦ brother ⊆ uncle
  ● ancestor ⊏ parent¹
Reasoning Examples — what you CAN’T do

Where to begin!

☞ Robust decidability largely due to tree model property
  ● For any consistent class there exists a tree (like) model

☞ No property constructors, e.g.:
  ● parent ◦ brother ⊆ uncle
  ● ancestor ⊃ parent+

☞ No variables, e.g.:
  ● Ulcer ∧ ∃hasLocation.?x ⊆ ∃hasLocation.(∃LiningOf.?x)
Reasoning Examples — what you CAN’T do

Where to begin!

☞ Robust decidability largely due to **tree model property**
   - For any consistent class there exists a tree (like) model

☞ No **property constructors**, e.g.:
   - parent ◦ brother ⊆ uncle
   - ancestor ⊆ parent^*

☞ No **variables**, e.g.:
   - Ulcer ∧ ∃ hasLocation.?x ⊆ ∃ hasLocation.(∃ LiningOf.?x)

☞ Only have **unary and binary predicates**
   - Can’t express (directly) \( P(x, y, z) \)
Reasoning Examples — what you CAN’T do

Where to begin!

☞ Robust decidability largely due to tree model property
  ● For any consistent class there exists a tree (like) model

☞ No property constructors, e.g.:  
  ● parent ◦ brother ⊆ uncle  
  ● ancestor ⊆ parent

☞ No variables, e.g.:  
  ● Ulcer ⊗ ∃hasLocation. ?x ⪯ ∃hasLocation.(∃LiningOf.?x)

☞ Only have unary and binary predicates  
  ● Can’t express (directly) \( P(x, y, z) \)

Language extensions may remove some of above limitations
Reasoning Examples — what you CAN’T do

Where to begin!

☞ Robust decidability largely due to **tree model property**
  - For any consistent class there exists a tree (like) model

☞ No **property constructors**, e.g.:
  - parent ° brother ⊨ uncle
  - ancestor ⊨ parent+

☞ No **variables**, e.g.:
  - Ulcer ∧ ∃hasLocation.?x ⊨ ∃hasLocation.(∃LiningOf.?x)

☞ Only have **unary and binary predicates**
  - Can’t express (directly) $P(x, y, z)$

Language extensions may remove some of above limitations

☞ But there is **no such thing as a free lunch**