AIFB

Intelligent Systems on the World Wide Web

OIL + DAML-Ont

Lecture Slides
Steffen Staab
Institute for Applied Computer Science and Formal Description Methods (AIFB)
Karlsruhe University

AIFB

Adding formal semantics to the Web
building on top of RDF Schema

Jeen Broekstra, Michel Klein, Stefan Decker, Dieter Fensel, Ian Horrocks

On-To-Knowledge project

AIFB

Context

On-To-Knowledge

IST project about content-driven knowledge management through evolving ontologies

http://www.ontoknowledge.org/

OIL = Ontology Inference Layer

http://www.ontoknowledge.org/oil

AIFB

Contents

- Semantic annotation: why, and how?
  - how: XML?
  - how: RDF(S)?
  - how: The W3C vision
- OIL
- extending RDF Schema
- Conclusion and summary

Semantic annotation: why

- Semantic annotations make the meaning **machine-accessible:**
  - Intelligent information brokering
  - Meaning-based searches
  - Prerequisite for many agent applications

Is XML sufficient for semantic annotation?

- XML: user definable and domain specific markup
- XML document is a labeled tree
- constraints on structure via DTD or XML Schema

```
<animal>
        <name>Tux</name>
        <species>penguin</species>
        <eats>fish</eats>
        ...
</animal>
```


Shortcomings of XML
(for our purposes)

XML makes no commitment on:

❶ Domain specific ontological vocabulary

❷ Ontological modeling primitives

⇨ requires pre-arranged agreement on ❶ & ❷

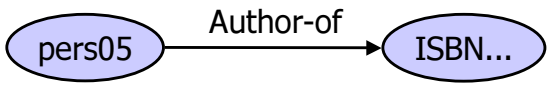
Only feasible for closed collaboration

- agents in a small & stable community
- pages on a small & stable intranet

not for sharable Web-resources

Is RDF(S) sufficient for semantic annotation?

- RDF provides metadata about Web resources
- Object -> Attribute-> Value triples



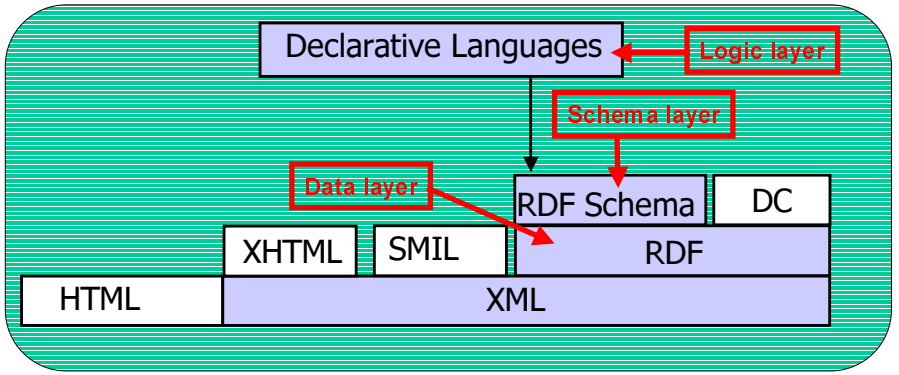
- RDF Schema
  - Defines vocabulary for RDF
  - Organizes this vocabulary in a typed hierarchy
    - `Class, subClassOf, type`
    - `Property, subPropertyOf`
    - `domain, range`

Conclusions about RDF(S)

- Next step up from plain XML:
  - (small) ontological commitment to modeling primitives
  - possible to define vocabulary
- However:
  - no precisely described meaning
  - no inference model

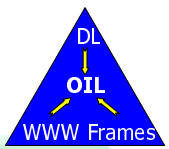
Semantic annotation: how

W3C's vision: The Semantic Web



OIL

- Based on standard frame languages (OKBC)
  - restricts & extends
- formalized by DL style logical constructs
- Still has frame "look and feel"
- Can still function as a basic frame language
- OIL language restricted:
  - to allow for reasoning support

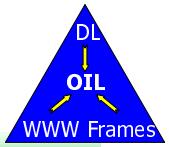


OIL (explained by example)

```
class-def animal                          % animals are a class
class-def plant                           % plants are a class
   subclass-of not animal                 % that is disjoint from animals
class-def tree
   subclass-of plant                      % trees are a type of plants
class-def branch
   slot-constraint is-part-of             % branches are parts of some tree
      has-value tree
      max-cardinality 1
class-def defined carnivore               % carnivores are animals
   subclass-of animal
   slot-constraint eats                   % that eat any other animals
      value-type animal
class-def defined herbivore               % herbivores are animals
   subclass-of animal, not carnivore      % that are not carnivores, and
   slot-constraint eats                   % they eat plants or parts of plants
      value-type plant or (slot-constraint is-part-of has-value plant)
```

OIL has a formal semantics

- Defined by mapping to expressive DL
  - slot-constraint eats has-value meat, fish
    =
    $\exists$ eats:meat $\cap$ $\exists$ eats:fish
- Mapping is used to provide reasoning support from a DL system (e.g., FaCT)



Extending RDF Schema

Goal

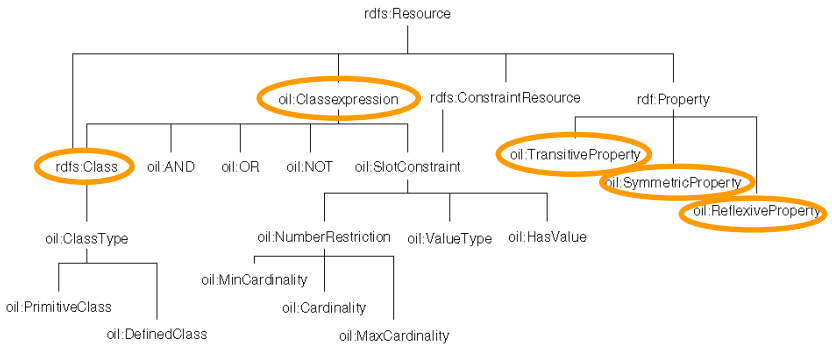
- make RDFS useable as ontology language
  - give RDF(S) precise semantics
  - extend RDF(S) with additional modeling primitives
- to facilitate semantically grounded metadata

Procedure

- formulate ontology language as RDF Schema document
  - using existing primitives as much as possible
  - placing additional primitives in the hierarchy of RDFS primitives

OIL as extension to RDFS (1)

- part of the is-a hierarchy of RDFS extension
- ontology language is defined in RDFS



OIL as extension to RDFS (3)

```
<rdfs:Class rdf:ID="herbivore">
  <rdf:type
      rdf:resource="http://www.ontoknowledge.org/#DefinedClass"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
  <rdfs:subClassOf>



  </rdfs:subClassOf>
</rdfs:Class>
```

AIFB

Using the extension: three levels

❶ OIL modeling primitives
slot-constraint, subclass-of, value-type,...
– RDF-S document which extends RDF-S

❷ a specific OIL ontology
animal, plant, herbivore, leaf
– RDF-S document (using ❶)

❸ instances of this ontology
"Mel the giraffe", "Tux the penguin"
– RDF expressions (uses ❶&❷)
– explicit metadata

AIFB

What did we gain?

- Any RDF agent can
  process OIL instances
- Any RDF-S agent can
  process OIL ontologies
- Any OIL-aware agent can
  exploit semantics & reasoning
  (and materialize the OIL derivations
  for use by OIL-ignorant RDF agents)

AIFB

Some more examples...

AIFB

OIL: Erweitert Frame-Sprachen

- Klassen können primitiv sein (notwendige Bedingungen)
  – elephant $\Rightarrow$ animal that has-colour grey
- oder definiert (notwendige und hinreichende Bedingungen)
  – vegetarian $\Leftrightarrow$ person who eats meat nor fish
- Klassen sind als Slot-Constraints zugelassen
  – slot-constraint eats has-value meat
    (eats some meat)
  – slot-constraint eats value-type meat
    (eats only meat)

OIL: Erweitert Frame-Sprachen

- Verwendung arbiträrer Klassennamen
  - slot-constraint eats value-type NOT (OR meat fish)
- Kardinalitätsrestriktion kann Klassennamen enthalten
  - slot-constraint eats max-cardinality 1 plant
- Die sub-slot-Relation wird unterstützt
  - daughter-of sub-slot of child-of
- Slot-Eigenschaften können spezifiziert werden
  - transitive (e.g., part-of)
  - symmetrical (e.g., connected-to)

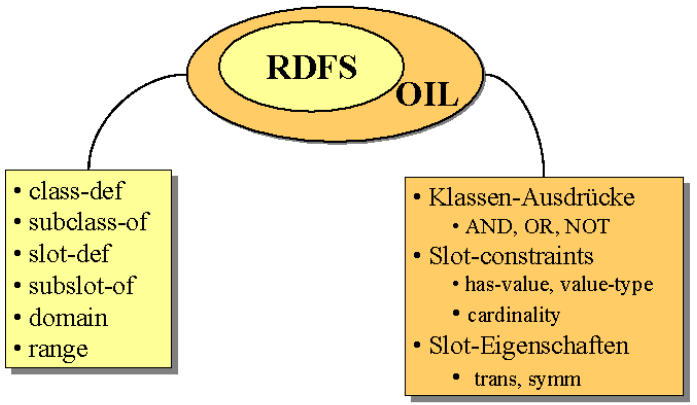
OIL: Einfaches Beispiel

```
class-def animal                    % animals are a class
class-def plant                     % plants are a class
   subclass-of NOT animal           % that is disjoint from animals
class-def tree
   subclass-of plant                % trees are a type of plants
class-def branch
   slot-constraint is-part-of       % branches are parts of some tree
      has-value tree
      max-cardinality 1
class-def defined carnivore         % carnivores are animals
   subclass-of animal
   slot-constraint eats             % that eat any other animals
      value-type animal
class-def defined herbivore         % herbivores are animals
   subclass-of animal, NOT carnivore % that are not carnivores, and
   slot-constraint eats             % they eat plants or parts of plants
      value-type plant OR (slot-constraint is-part-of has-value plant)
```


OIL als RDFS-Erweiterung

```
<rdfs:Class rdf:ID="herbivore">
  <rdf:type
      rdf:resource="http://www.ontoknowledge.org/#DefinedClass"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
  <rdfs:subClassOf>
     <oil:NOT>
        <oil:hasOperand rdf:resource="#carnivore"/>
     </oil:NOT>
  </rdfs:subClassOf>
</rdfs:Class>
```


OIL als RDFS-Erweiterung



OIL in XML:

Für OIL gibt es ein DTD, ein XML-Schema und eine Abbildung auf RDFS.

```
<slot-def>
  <slot-name = "has-component"/>
  <inverse> <slot-name = "is-component-of"/> </inverse>
  <properties> <transitive/> </properties>
</slot-def>
<class-def> <class-name= "nucleic-acid"/> </class-def>
<class-def>
  <class-name= "rna"/>
  <subclass-of> <class name = "nucleic-acid"/> </subclass-of>
  <slot-constraint>
    <slot-name = "has-backbone"/>
    <value-type> <class name= "ribophosphate" </value-type>
  </slot-constraint>
</class-def>
```


OIL: Ontologie-Metadaten (gemäß Dublin Core)

Ontology-container
title "macromolecule fragment"
creator "robert stevens"
subject "macromolecule generic ontology"
description "example for a tutorial"
description.release "1.0"
publisher "R Stevens"
type "ontology"
formal "pseudo-xml"
identifier "http://www.ontoknowledge.org/oil/oil.pdf"
source "http://img.cs.man.ac.uk/ismb00/mmexample.pdf"
language "OIL"
language "en-uk"
relation.haspart "http://www.ontoRus.com/bio/mmole.onto"

OIL: Primitive Bio-Ontologie-Definition

slot-def has-backbone
  inverse is-backbone-of
slot-def part-of
  inverse is -part-of
  properties transitive

class-def rna
  subclass-of nucleic-acid
  slot-constraint has-backbone
    value-type ribophosphate

class-def ribophosphate
class-def deoxyribophosphate
  subclass-of NOT ribophosphate

OIL: Definierte Klassen der Bio-Ontologie

class-def defined dna
  subclass-of nucleic-acid
  slot-constraint has-backbone
    value-type deoxyribophosphate
class-def dna
  subclass-of NOT rna

class-def defined catalyst
  subclass-of macromolecule
  slot-constraint promotes
    has-value reaction

class-def defined enzyme
  subclass-of protein, catalyst

AIFB

OIL: Definierte Klassen der Bio-Ontologie

class-def defined mitochondrial
  subclass of location
  slot-constraint cellularlocation
    cardinality 1 ((**has-value** mitochondrion) OR
    (**slot-constraint** part-of **has-value** mitochondrion))

class-def defined succinate-dehydrogenase
  subclass of enzyme
  slot-constraint promotes
    value-type oxidation
  slot-constraint cellularlocation
    cardinality 1 (**has-value** (**slot-constraint** part-of **has-value**
      mitochondrion)

AIFB

OIL: Beispiel einer Ontologie für Drucker (Teil 1)

class-def Product
slot-def Price
  domain Product
slot-def ManufacturedBy
  domain Product
class-def PrintingAndDigitalImagingProduct
  subclass-of Product
class-def HPProduct
  subclass-of Product
  slot-constraint ManufacturedBy
    has-value "Hewlett Packard"
class-def Printer
  subclass-of PrintingAndDigitalImagingProduct
slot-def PrinterTechnology
  domain Printer
slot-def Printing Speed
  domain Printer
slot-def PrintingResolution
  domain Printer
class-def PrinterForPersonalUse
  subclass-of Printer
class-def HPPrinter
  subclass-of HPProduct and Printer

AIFB

OIL: Beispiel einer Ontologie für Drucker (Teil 2)

class-def LaserJetPrinter
  subclass-of Printer
  slot-constraint PrintingTechnology
    has-value "Laser Jet"
class-def HPLaserJetPrinter
  subclass-of LaserJetPrinter and HPProduct
class-def HPLaserJet1100Series
  subclass-of HPLaserJetPrinter and PrinterForPersonalUse
  slot-constraint PrintingSpeed
    has-value "8 ppm"
  slot-constraint PrintingResolution
    has-value "600 dpi"
class-def HPLaserJet1100se
  subclass-of HPLaserJet1100Series
  slot-constraint Price
    has-value "$479"
class-def HPLaserJet1100xi
  subclass-of HPLaserJet1100Series
  slot-constraint Price
    has-value "$399"

AIFB

OIL in OIL

class-def oil-ontology-definition
  documentation "a set of definitions about import and definitions"
  slot-constraint oil-import
    value-type string
    min-cardinality 0 string
  slot-constraint oil-definition
    value-type ((oil-class-def or oil-slot-def) or oil-axiom)

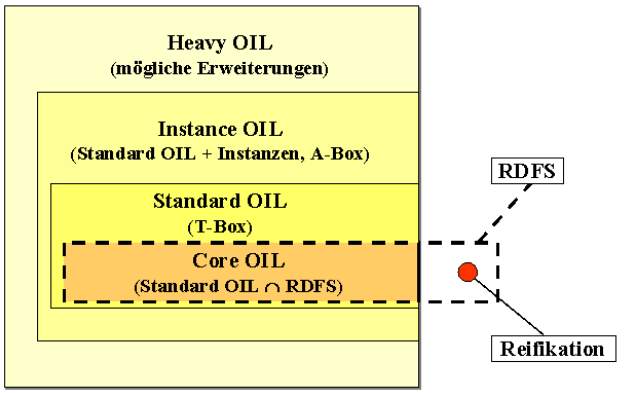
class-def oil-class-def
  documentation "A class definition associates a class name with a class description"
  slot-constraint oil-type
    value-type ((equal "primitive") or (equal "defined"))
  slot-constraint oil-name
    value-type string
  slot-constraint oil-documentation
    value-type string
    min-cardinality 0 string
  slot-constraint has-slot-constraint
    value-type class-expression
    min-cardinality 0 class-expression

OIL in OIL

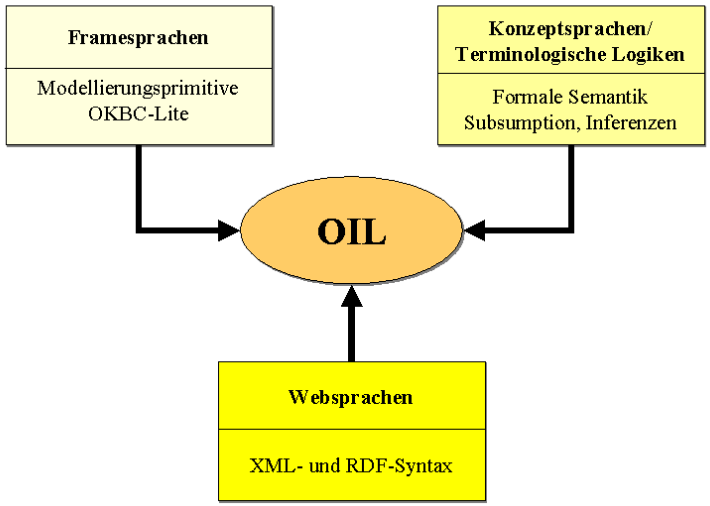
class-def oil-or
documentation "A list of two or more class expressions that is to be treated as a disjunction"
  subclass-of class-expression
  slot-constraint has-operand
    value-type class-expression
    cardinality 2 class-expression

class-def oil-slot-def
  documentation "a slot definition associates a slot name with a slot description"
  slot-constraint oil-name
    value-type string
  slot-constraint oil-documentation
    value-type string
    min-cardinality 0 string
  slot-constraint oil-subslot-of
    value-type slot-expression
    min-cardinality 0 slot-expression
  slot-constraint oil-inverse
    value-type slot-expression
    min-cardinality 0 slot-expression
  slot-constraint oil-properties
    value-type (((equal "transitive") or (equal "symmetric")) or (equal "functional"))

OIL: Ontology Inference Layer / Ontology Interchange Language



OIL führt drei Sprachfamilien zusammen



DAML: DARPA Agent Markup Language

DAML-ONT ist in RDF geschrieben, das selbst in XML codiert ist und XML Namensräume verwendet.

Zunächst werden in der hier behandelten DAML-Beispielontologie drei Namensräume definiert:

```
<rdf:RDF
  xmlns:rdf    ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns        ="http://www.daml.org/2000/10/daml-ont#"
  xmlns:daml="http://www.daml.org/2000/10/daml-ont#"
  >
```

1) rdf-Präfix mit Verweis auf die Standarddefinitionen für RDF.

2) Unpräfigierte Elementnamen beziehen sich auf die Standard-Ontologie von DAML.

3) Alternativ können diese Elemente auch mit dem daml-Präfix versehen werden.

DAML: DARPA Agent Markup Language

Zunächst wird erklärt, daß nun eine spezielle Ontologie definiert wird. Dies ist eine "leere" Assertion, da der Rest des RDF-Dokuments gerade den Inhalt darstellt.

```
<Ontology about="">
  <versionInfo>$Id: daml-ex.daml,v 1.2 2000/10/07 03:21:17 connolly Exp
  </versionInfo>
  <comment>An example ontology</comment>
  <imports resource="http://www.daml.org/2000/10/daml-ont"/>
</Ontology>
```

Es folgt Versioneninformation (versionInfo) und Information über Ontologien, die bei der Definitionen der neuen Ontologie verwendet werden sollen (imports).

DAML: DARPA Agent Markup Language

Definitionen von Klassen und Eigenschaften in DAML:

```
<Class ID="Animal">
  <label>Animal</label>
  <comment>This class of animals is illustrative of a number of
           ontological idioms.</comment>
</Class>
```

Label führt Bezeichner ein, z.B. zur graphischen Darstellung der Ontologie. Comment und Label wird in DAML nicht logisch interpretiert. Mit ID wird gewährleistet, daß die Klasse Animal extern über eine URI und #Animal referenziert werden kann.

```
<Class ID="Male">
  <subClassOf resource="#Animal"/>
</Class>
```

Male wird als Subklasse von Animal eingeführt.

DAML: DARPA Agent Markup Language

Mit den disjointFrom-Tag können ähnlich wie in Konzeptsprachen Disjunktheitsklassen definiert werden:

```
<Class ID="Female">
  <subClassOf resource="#Animal"/>
  <disjointFrom resource="#Male"/>
</Class>
```

Eine Eigenschaft ist eine binäre Relation wie Rollen in Konzeptsprachen.

```
<Property ID="parent">
  <domain resource="#Animal"/>
  <cardinality>2</cardinality>
</Property>
```

Es wird festgelegt, daß der Definitionsbereich der Relation Animals sind und es jeweils zwei Eltern gibt (cardinatity-Tag).

DAML: DARPA Agent Markup Language

Einschränkungen von Relationen:

```
<Class ID="Person">
  <subClassOf resource="#Animal"/>
  <restrictedBy>
    <Restriction>
      <onProperty resource="#parent"/>
      <toClass resource="#Person"/>
    </Restriction>
  </restrictedBy>
</Class>
```

Der Bildbereich der Relation Parent wird für Personen auf Personen eingeschränkt:

```
<Property ID="father">
  <subProperty resource="#parent"/>
  <range resource="#Man"/>
  <cardinality>1</cardinality>
</Property>
```

Vater wird als Subrelationen (Subrollen in KL-ONE) von Eltern eingeführt, wobei es immer nur einen Vater gibt.

DAML: DARPA Agent Markup Language

Notationsvariante für Relationen mit Kardinalität 1 (funktionale Rollen):

```
<UniqueProperty ID="mother">
  <subProperty resource="#parent"/>
  <range resource="#Woman"/>
</UniqueProperty>
```

Inverse Relationen:

```
<Property ID="child">
  <inverseOf resource="#parent"/>
</Property>
```

Vorfahre und Nachfahre als transitive Relationen:

```
<TransitiveProperty ID="ancestor">
  <label>ancestor</label>
</TransitiveProperty>

<TransitiveProperty ID="descendant"/>
```


DAML: DARPA Agent Markup Language

Synonyme können in der Ontologie eingeführt werden:

```
<Property ID="mom">
  <equivalentTo resource="#mother"/>
</Property>
```

Es kann auch Kardinalitätsbereich angegeben werden (0 oder 1 Beruf):

```
<Property ID="occupation">
  <maxCardinality>1</maxCardinality>
</Property>
```

Mit dem complementOf-Tag wird eine anonyme Klasse gebildet – die Komplementmenge aller Personen:

```
<Class ID="Car">
  <subClassOf>
    <Class>
      <complementOf resource="#Person"/>
    </Class>
  </subClassOf>
</Class>
```


DAML: DARPA Agent Markup Language

Multiple Vererbung und Konjunktion von Konzeptklassen:

```
<Class ID="Man">
  <subClassOf resource="#Person"/>
  <subClassOf resource="#Male"/>
</Class>

<Class ID="Woman">
  <subClassOf resource="#Person"/>
  <subClassOf resource="#Female"/>
</Class>
```

Person als disjunkte Vereinigung von Männern und Frauen (führt zu Überdeckung wie in Konzeptsprachen) durch Erweiterung der Person Definition (about=):

```
<Class about="#Person">
  <disjointUnionOf parseType="daml:collection">
    <Class about="#Man"/>
    <Class about="#Woman"/>
  </disjointUnionOf>
</Class>
```


DAML: DARPA Agent Markup Language

Definition einer Instanz von Person:

```
<Person ID="Adam">
  <label>Adam</label>
  <comment>Adam is a person.</comment>
</Person>
```

Eine Person hat eine bestimmte Größe (Eigenschaft):

```
<Property ID="height">
  <domain resource="#Person"/>
  <range resource="#Height"/>
</Property>
```

Die Größe wird durch eine extensional definierte Menge von Größenangaben spezifiziert:

```
<Class ID="Height">
  <oneOf parseType="daml:collection">
    <Height ID="short"/>
    <Height ID="medium"/>
    <Height ID="tall"/>
  </oneOf>
</Class>
```

DAML: DARPA Agent Markup Language

Durch Wertrestriktion können neue Klassen spezifiziert werden:

```
<Class ID="TallThing">
 <restrictedBy>
  <Restriction>
   <onProperty resource="#height"/>
   <toValue resource="#tall"/>
  </Restriction>
 </restrictedBy>
</Class>
```

Durch eine Durchschnittsbildung können neue Klassen aus der Kombination von bereits definierten Klassen spezifiziert werden:

```
<Class ID="TallMan">
 <intersectionOf parseType="daml:collection">
  <Class about="#TallThing"/>
  <Class about="#Man"/>
 </intersectionOf>
</Class>
```

</rdf:RDF> Abschluß der Ontologie-Definition über RDF

DAML: DARPA Agent Markup Language

Analog zu den Konzeptsprachen kann eine modell-theoretische Semantik für DAML-ONT definiert werden.

Die Semantik wird durch eine Menge von Axiomen in Prädikatenlogik erster Ordnung, die mithilfe von KIF (Knowledge Interchange Format ANSI Vorschlag) definiert sind, formal spezifiziert.

```
<Property ID="disjointWith">
<label>disjointWith</label>
<comment>
     for disjointWith(X, Y) read: X and Y have no members in common.
</comment>
<domain resource="#Class"/>
<range resource="#Class"/>
</Property>
```

```
(Property disjointWith)
(domain disjointWith Class)
(range disjointWith Class)
```

```
(<=> (disjointWith ?c1 ?c2)
(and (exists ?x (or (type ?x ?c1) (type ?x ?c2)))
(not (exists ?x (and (type ?x ?c1) (type ?x ?c2)))))
```


DAML: DARPA Agent Markup Language

Definition einer Liste von disjunkten Objekten:

```
<Class ID="Disjoint">
<label>Disjoint</label>
<subClassOf resource="#List"/>
<comment>for type (L, Disjoint) read: the classes in L are pairwise disjoint.
        .e. if type (L, Disjoint), and C1 in L and C2 in L, then
   disjointWith(C1, C2).
</comment>
</Class>
```

KIF:

```
(subClassOf Disjoint List)
```

```
(=> (Disjoint ?l) (Item ?c ?l) (Class ?c))
```

```
(=> (Disjoint ?l)
(Item ?c1 ?l)
(Item ?c2 ?l)
(~= ?c1 ?c2)
(DisjointWith ?c1 ?c2))
```


DAML: DARPA Agent Markup Language

```
<Property ID="unionOf">
<label>unionOf</label>
<comment>
     for unionOf(X, Y) read: X is the union of the classes in the list Y;
     i.e. if something is in any of the classes in Y, it's in X, and vice versa.
</comment>
<domain resource="#Class"/>
<range resource="#List"/>
</Property>
```

KIF:

```
(Property unionOf)
(Domain unionOf Class)
(Range unionOf List)
```

```
(=> (unionOf ?c ?l) (item ?x ?l) (Class ?x))
```

```
(<=> (unionOf ?c1 ?l)
(<=> (type ?x ?c1)
(exists ?c2 (and (item ?c2 ?l) (type ?x ?c2)))))
```

DAML: DARPA Agent Markup Language

Definition der disjunkten Vereinigung:

```
<Property ID="disjointUnionOf">
<label>disjointUnionOf</label>
<domain resource="#Class"/>
<range resource="#List"/>
<comment>
    for disjointUnionOf(X, Y) read: X is the disjoint union of the classes in
    the list Y: (a) for any c1 and c2 in Y, disjointWith (c1, c2), and (b) i.e. if
something is in any of the classes in Y, it's in X, and vice versa.
</comment>
</Property>
```

KIF:

```
(Property disjointUnionOf)
(Domain disjointUnionOf Class)
(Range disjointUnionOf Disjoint)
(<=> (disjointUnionOf ?c ?l) (and (unionOf ?c ?l) (Disjoint ?l)))
```


DAML: DARPA Agent Markup Language

Transitive und funktionale Eigenschaften:

```
<Class ID="TransitiveProperty"/>
(subClassOf TransitiveProperty Property)
```

KIF:

```
(<=> (TransitiveProperty ?p)
(forall (?x ?y ?z)
(=> (holds ?p ?x ?y) (holds ?p ?y ?z) (holds ?p ?x ?z))))
```

```
<Class ID="UniqueProperty">
<label>UniqueProperty</label>
<subClassOf resource="#Property"/>
</Class>
```

KIF:

```
(subClassOf UniqueProperty Property)
(<=> (UniqueProperty ?p) (maxCardinality ?p 1))
```


Future

- DAML+OIL
- DAML-L(ogic)
  - Including Rules
- DAML-S(ervice)
  - Primitives for intelligent services
    - ⇒ **http://www.aifb.uni-karlsruhe.de/Lehrangebot/Winter2001-02/ebiz+iw/**

Summary

- to enable intelligent information handling, machine-understandable semantics are needed
- advantages of our approach
  - reuse of modeling primitives
  - conform W3C view of the world
  - added benefits (from OIL):
    - **reasoning support**
    - **formal semantics**
- Full schema available
  - http://www.ontoknowledge.org/oil/rdf-schema