MANCHESTER 1824

The University of Manchester

Description Logic Reasoning

COMP62342
Sean Bechhofer
sean.bechhofer@manchester.ac.uk

MANCHESTER 1824

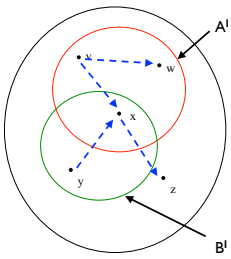
The University of Manchester

Inference

- Ontologies provide
  - **Vocabulary** that describes a domain
  - **Assumptions** about the ways in which that vocabulary should be interpreted
- What can we then *infer* from that information?
  - In particular, what inferences can be drawn from the assumptions that we've expressed?

DL Semantics

- Recall that our semantics was defined in terms of *Interpretations*
  - Domain of discourse Δ
  - Function I mapping:
  - Individuals names x, y, z to elements of Δ
  - Class names A, B, C to subsets of Δ
  - Property names R, S, to sets of pairs of elements of Δ



DL Semantics

- Interpretations then extended to cover concept expressions
  - $(A \sqcap B)^I = A^I \cap B^I$
- Interpretation is a *model* of an axiom A iff the interpretation of the axiom holds
  - $I \models A \sqsubseteq B$ iff $A^I \subseteq B^I$
- Interpretation is a *model* of an ontology O iff it is a model of all the axioms in O

Note use of logical ("German syntax") here rather than Manchester Syntax.

MANCHESTER 1824

The University of Manchester

Inference

- What can we *infer* from an Ontology O?
  - And what do we mean by *infer*?

- The semantics describe the conditions under which an interpretation is a model of a Ontology O.
- It can be the case that due to the constraints that O places on the interpretations, there are consequences that also hold in all the interpretations.
- Recall, an Ontology doesn't define a single model, it is a set of constraints that define a set of possible models
  - No constraints (empty Ontology) means any model is possible
  - More constraints means fewer models
  - Too many constraints may mean no possible model (inconsistent Ontology)

5

MANCHESTER 1824

The University of Manchester

Basic Inference Problems

- Subsumption
  - $C \sqsubseteq_O D$ iff $C^I \subseteq D^I$ in *all* models I of O
- Equivalence
  - $C \equiv_O D$ iff $C^I = D^I$ in *all* models I of O
- Satisfiability
  - $C \not\equiv_O \bot$ iff C^I non empty in *some* model I of O
- Instantiation
  - $i \in_O C$ iff $i \in C^I$ in *all* models I of O
- Consistency
  - O consistent iff there is *at least* one model I of O
- Coherency
  - O coherent iff all names classes are satisfiable
- Problems reducible to satisfiability:
  - e.g., $C \sqsubseteq_O D$ iff $(C \sqcap \neg D)$ not satisfiable w.r.t. O

6

MANCHESTER 1824

The University of Manchester

Example Inferences

- O = {B ⊑ A, C ⊑ B}
  - C $\sqsubseteq_O$ A
- O = {C ⊑ A ⊓ B }
  - C $\sqsubseteq_O$ A
- O = {}
  - A $\sqsubseteq_O$ A ⊔ B
- O = {C ≡ ∃R.A, B ⊑ A }
  - ∃R.A $\sqsubseteq_O$ C
- O = {B ⊑ A, x:B}
  - x $\in_O$ B
- O = {C ⊑ A ⊓ B }
  - C $\not\equiv_O \bot$
- O = {C ⊑ A, C ⊑ ¬A}
  - C $\equiv_O \bot$
  - O incoherent
- O = {C ⊑ A, C ⊑ ¬A, x:C}
  - O inconsistent

7

MANCHESTER 1824

The University of Manchester

Consistency and Unsatisfiability

- Note the difference between *class* satisfiability and *ontology* consistency
- A class C is unsatisfiable if there are no models such that its interpretation is non-empty
- An Ontology O is inconsistent if there are no models of O
- A consistent Ontology *may* contain unsatisfiable classes.
- O = {C ⊑ A ⊓ B, D ⊑ C ⊓ ¬ B }
  - D unsatisfiable, but models exists thus O is consistent…
- O = {C ⊑ ∀R.¬A, x:C, y:A, <x,y>:R}
  - Inconsistent Ontology

8

MANCHESTER 1824

The University of Manchester

Why are these useful?

- Subsumption: check knowledge is correct
  - Build classification hierarchies of primitive (named) classes
- Equivalence: check knowledge is minimally redundant
- Satisfiability: check knowledge is meaningful
- Instantiation: check if individual i instance of class C.
  - Supporting query.

9

MANCHESTER 1824

The University of Manchester

Structural Approaches

- Early implementations used *structural* approaches
- E.g. to check subsumption
  1. Normalise expressions
  2. Compare the structure of the expressions to see if there is "overlap".
- This is effective, but hard to get complete results, particularly in the face of complex axioms (or GCIs as they are sometimes known).
- An alternative is to use an approach based on the underlying semantics, e.g. *tableaux*.

10

MANCHESTER 1824

The University of Manchester

Tableaux Algorithms: Basics

- Tableaux algorithms used to test satisfiability
- Try to build tree-like model I of input class C
- Work on classes in negation normal form
  - Rewrite and push in negation using de Morgan's laws
  - E.g. $\neg\exists R.C$ to $\forall R.\neg C$
- Break down C syntactically, inferring constraints on elements of I
- Decomposition uses tableau rules corresponding to constructors in the logic (e.g $\sqcap$, $\exists$)
  - Some rules are nondeterministic, e.g. they involve some choice
    - $\sqcup$, $\cdot$
  - In practice, this means search.

11

MANCHESTER 1824

The University of Manchester

Tableaux Algorithms: Basics

- Try and build a "completion tree" by applying rules
- Stop when a clash occurs or when no more rules are applicable.
- Blocking (cycle check) used to guarantee termination
- Returns "C is consistent" iff C is consistent
  - Tree model property

12

Tableaux Algorithms: Details

- Work on tree T representing model I of class C
  - Nodes x,y represent elements of domain Δ
  - Nodes labelled with L(x), sub-expressions of C
  - Edges represent role-successorships between elements of Δ
- T initialised with single root node labelled {C}
- Tableau rules repeatedly applied to node labels.
  - Extend labels of a node or extend/modify the tree structure
  - Rules can be blocked, e.g. if a predecessor node has superset label
  - Nondeterministic rules mean we may need to search for possible extensions
- T contains a Clash if there is an obvious contradiction in some node label
  - E.g. $\{A, \neg A\} \subseteq L(x)$ for some class A and node x

13

Tableaux Algorithms: Details

- T fully expanded if no rules are applicable
- C satisfiable iff fully expanded clash-free tree T found
  - There is then a correspondence between T and a model of C (see later remarks regarding completeness)
- Thus the tableaux algorithm helps us by finding a witness for the consistency of C
  - There is some model.

14

MANCHESTER 1824

The University of Manchester

ALC

- Propositional constructors
  - ⊓, ⊔, ¬
- Role Quantifiers
  - ∃, ∀
- Top and Bottom
  - ⊤, ⊥

- Concept names, ⊤, ⊥ are Concepts. Role names are Roles
- For C, D concepts and R a role, the following are Concepts:
  - ¬ C
  - C ⊓ D
  - C ⊔ D
  - ∃ R.C
  - ∀ R.C

15

MANCHESTER 1824

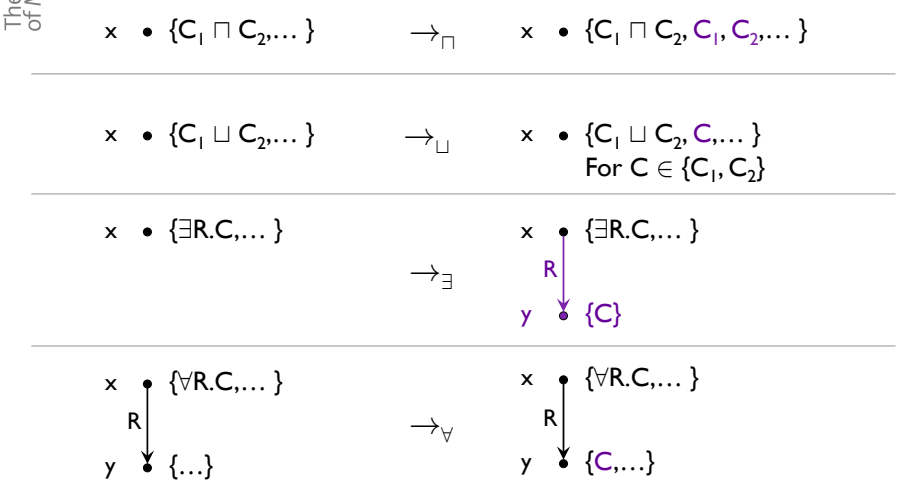
The University of Manchester

ALC and OWL

ALC	OWL
⊓	and
⊔	or
¬	not
∃	some
∀	only
⊤	thing
⊥	nothing

16

Tableaux Rules for ALC



17

Algorithm Examples

- Test the satisfiability of

$$\exists R.A \sqcap \forall R.B$$

$$\exists R.A \sqcap \forall R.\neg A$$

$$\exists R.A \sqcap \forall S.\neg A$$

$$\exists R.(A \sqcup \exists R.B) \sqcap \forall R.\neg A \sqcap \forall R.(\forall R.\neg B)$$

18

MANCHESTER 1824

The University of Manchester

Is it right?

- How do we know whether our algorithm is doing the "right thing"?
  - And what is the "right thing"?
- Soundness and Completeness help us characterise this.
  - Soundness: we get correct answers
  - Completeness: we get all the answers
- For our tableaux algorithm
  - Soundness: if the algorithm says that C is satisfiable, then it is (according to the semantics)
  - Completeness: if C is satisfiable (according to the semantics) then the algorithm will tell us this

19

MANCHESTER 1824

The University of Manchester

Termination

- Given a concept expression C, our algorithm will terminate
- Informal argument:
  - Rules (other than $\rightarrow_\forall$) are never applied twice on the same label
  - The $\rightarrow_\forall$ rule is never applied to node N more that n times where n is the number of direct successors of N.
  - Each rule application on a label C adds labels D such that D is a strict sub-expression of C

20

MANCHESTER 1824

The University of Manchester

Soundness

- If, given a concept description C, the algorithm terminates with a clash-free completion tree, then C is satisfiable
- Informal argument
  - Given the clash-free completion tree, we can produce an interpretation where C is non-empty

21

MANCHESTER 1824

The University of Manchester

Completeness

- For completeness, we need to show that given a satisfiable concept expression C, if we start the tableaux with C, then we will arrive at a fully expanded clash-free tree
- Informal argument
  - As C is satisfiable, we know there is an interpretation I where C^I is non-empty.
  - We can use this interpretation to guide the construction of the tableaux – in particular guiding choices.

22

Satisfiability w.r.t Axioms

- Our basic algorithm just operates on class expressions and doesn't consider any axioms
- For each axiom $C \sqsubseteq D$ in O, add $\neg C \sqcup D$ to every node label
  - Can rewrite Ontology in terms of $\sqsubseteq$
- Potentially very expensive!
  - Adding a disjunction to *every* node in the graph

23

Unfolding

- *Unfolding* allows us to deal with particular forms of Ontology.
- Consider a Ontology O that only contains definitions
  - E.g. $C \equiv D$ or $C \sqsubseteq D$, where C is a concept name.
- For any concept A occurring in D, we say A directly uses D and define uses as the transitive closure of directly uses.
- O contains a *cycle* if there is an atomic concept that uses itself.
  - $\{A \sqsubseteq B, B \sqsubseteq C, C \equiv D\}$
  - $\{A \sqsubseteq B, B \equiv \exists R.C, C \sqsubseteq A\}$
- If O is acyclic, we can *expand* and *unfold* the Ontology.

24

Unfolding

- To test satisfiability of concept description C w.r.t. acyclic Ontology O
  1. For any axiom
     $$B \equiv A$$
  – Replace all occurrences of B in C with A.
  2. For any axiom
     $$B \sqsubseteq A$$
  – Replace all occurrences of B in C with B' $\sqcap$ A, where B' is a new concept name.
  3. Then proceed with tableaux as normal on the new description.

25

Unfolding Examples

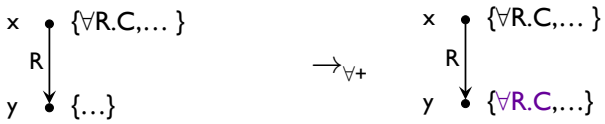
- Test satisfiability of
  $$\neg (\neg A \sqcup \exists R.C)$$

1. w.r.t the axioms
   $$\{A \sqsubseteq C \sqcap \exists R.C\}$$
2. w.r.t. the axioms
   $$\{A \equiv \neg C\}$$

26

Tableau Rule for Transitive Roles

- We can also consider ALC plus transitive roles
  - i.e. allowing assertions about the transitivity of a role (rather than allowing us to talk about the transitive closure
- This then requires an additional rule for transitive role R



- No longer naturally terminating (e.g. if C = ∃R.⊤)
- We need a blocking strategy
  - Simple blocking is enough for ALC + transitive roles
  - Do not expand the node label if the ancestor has superset label
  - Need more for more expressive logics.

27

Algorithm Examples

- Test the satisfiability of

$$\exists S.C \sqcap \forall S.(\neg C \sqcup \neg D) \sqcap \exists R.C \sqcap \forall R(\exists R.C)$$

Where R is a transitive role

28

MANCHESTER 1824

The University of Manchester

More Expressive DLS

- Basic technique can be extended to deal with
  - Role inclusion axioms
  - Number restrictions
  - Inverse Roles
  - Concrete domains
  - Aboxes
- Extend expansion rules and use more sophisticated blocking strategy.
- Forest instead of tree for Individual Facts
  - Root nodes correspond to individuals in Ontology.

29

MANCHESTER 1824

The University of Manchester

Scalability

- Reasoning with DL languages is hard
  - Ontologies on the web may grow large
  - Particularly with Instance data.
- Space usage
  - Storage required for tableaux datastructures
  - Rarely serious problem in practice
  - But problems with inverse roles and cyclical Ontologies
- Time usage
  - Non-deterministic rules lead to search
  - Serious problem in practice
  - Mitigated by
    - Careful choice of algorithm
    - Highly optimised implementations

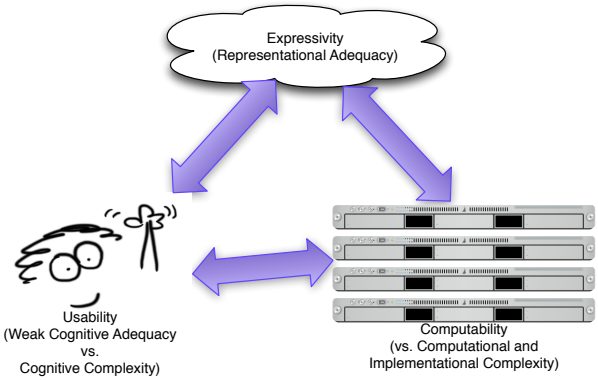
30

Choice of Algorithm

- Transitive roles rather than transitive closure
  - Deterministic expansion of $\exists R.C$ even when R in R+
  - Relatively simple blocking conditions
- Direct algorithm/implementation instead of encodings
  - GCI axioms can be used to encode additional operators/axioms
  - E.g. domain and range
    - (domain R C) $\equiv \exists R.\top \sqsubseteq C$
  - But even simple domain encoding yields terrible performance with large numbers of roles.

31

Trade Offs: The Design Triangle



32

Cognitive Adequacy

- Strong Cognitive Adequacy
  - A KR is SCA if it is "a (psychologically valid) cognitive model of [a human's] knowledge" (Strube, 1992)
    - If strong adequacy is claimed, the system is supposed to function like a human expert, at least in a circumscribed domain. In short, strongly adequate systems employ the very same principles of cognitive functioning as human experts do
- Weak Cognitive Adequacy
  - A KR is WCA if it is "ergonomic and user-friendly"
    - Note, however that the system may differ considerably from the experts (whose knowledge it attempts to represent) and from its users (if those are difference from the expert group). Still, the system tries to give users a comfortable feel, which may be achieved through symbols or words familiar to the user.

Gerhard Strube, The Role of Cognitive Science in Knowledge Engineering, 1992

33

Tradeoffs

- Syntax
  - How do we write things down?
- Expressivity
  - Ability of the language to distinguish between different concrete situations
  - If suitable to our needs, a formalism (or KR) is representationally adequate
- Computational Complexity
  - Reasoning
  - How hard is it to work with?
    - Theoretical Complexity
  - Implementational Complexity
    - How hard is it to produce a production quality implementation
- Cognitive Complexity
  - Focus on Weak Cognitive Adequacy i.e., Usability
  - How hard is it to understand or comprehend?
  - How much effort does it take to express something?
- A good KR (or KR formalism) achieves a good balance of all of these for most of its uses, most of the time

General desiderata:
- Clarity of specification
- Expressivity
- Usability
- Computability

34

MANCHESTER 1824

The University of Manchester

A Reasoning Perspective

- What expressivity do you need?
- What are your core service?
- What are the key services?
- Are you interactive or not?
- What's the scale you need to deal with?
  - And other performance characteristics
- What do you know about implementation?

- May neglect
  - Many surface syntax issues
  - Non logical aspects of the language
  - Cognitive complexity

35

MANCHESTER 1824

The University of Manchester

Services

- Core
  - Satisfiability
  - Consistency Checking
- Key
  - Entailment
  - Classification (atomic subsumptions)
    - Atomic class satisfiability
  - Instantiation
  - Query/ASK

- Querying
  - Basic logical inference services insufficient
  - DB style query languages
  - Supporting applications
- Explanation
  - *Why* do concepts subsume?
  - Supporting ontology design process
- Non-Standard Inferences
  - Least Common Subsumer
  - Matching
  - Supporting ontology design process

36

MANCHESTER 1824

The University of Manchester

Extra Logical Services

- It's not just about logic!
  - We also need services that are not directed related to the underlying formal semantics of the representation
- Annotation Services
  - Associating information with concepts.
  - Facilitating the use of an ontology within an application
  - "Conceptual Coatrack"
- Lexical Services
  - Associating words, terms or symbols with concepts
  - Facilitating understanding and use in applications
  - Rendering definitions

37

MANCHESTER 1824

The University of Manchester

Summary

- Tableaux Reasoning provides implementations for basic inference problems
  - Satisfiability
  - Subsumption
  - Classification
- Tableaux rules applied to try and build a treelike model of a concept (and thus demonstrate satisfiability)
- Further Reading:
  - Baader et.al. *The Description Logic Handbook*, Cambridge University Press, 2003

38