Knowledge Representation in PowerLoom Overview, Features and Examples

USC/ISI Loom KR&R Group

Overview

- Knowledge Representation (KR) Background
  - Evolution and Issues
- PowerLoom® Language
  - Concept and Relation Language
  - Assertions
  - Rules
  - Example
- PowerLoom Application
- Conclusion

- 1) The idea of descriptive logics and how they differ from, say, systems like prolog.
- 2) A little on the evolution of PowerLoom.
- 3) Details of Powerloom: The Concept/Relation language, Assertions, retrievals, open/closed world semantics
- 4) Rules: Forward and Backward chaining, the many ways to express rules. How to invoke rules explicitly.
- 5) Classification: What it is and how it works in Powerloom. Do the rabies example (it's on my website at: http://www-scf.usc.edu/~csci561a/slides/rabies.plm
- 6) How a PowerLoom application looks (especially one written in Java)

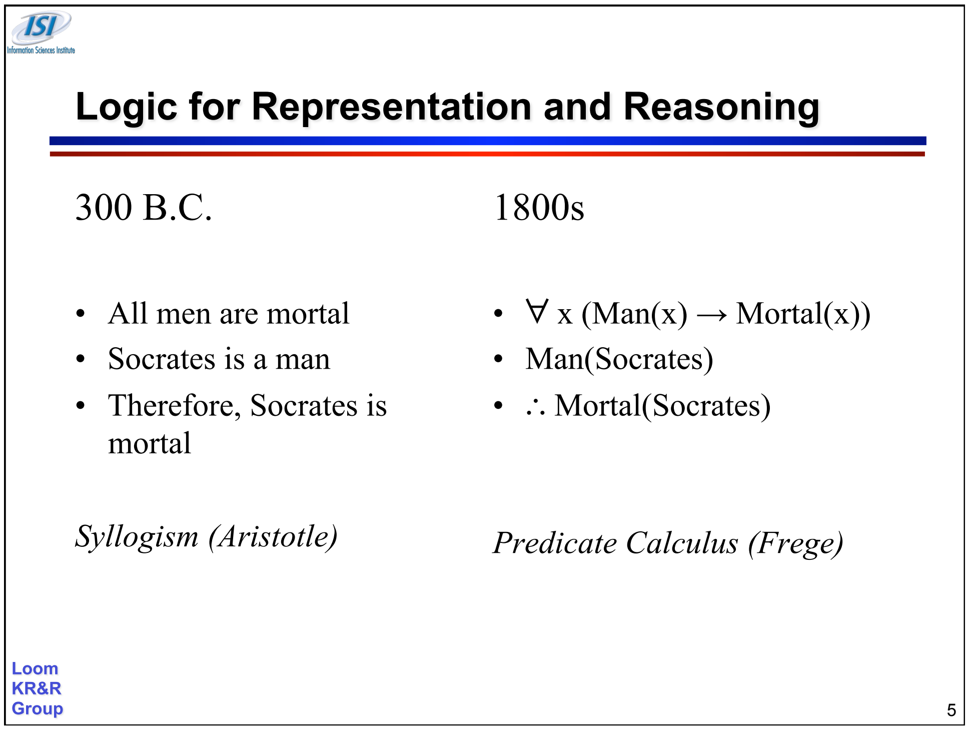
Loom
KR&R
Group

3

Knowledge Representation Background

Loom
KR&R
Group

4



Logic for Representation and Reasoning

300 B.C.

- All men are mortal
- Socrates is a man
- Therefore, Socrates is mortal

Syllogism (Aristotle)

1800s

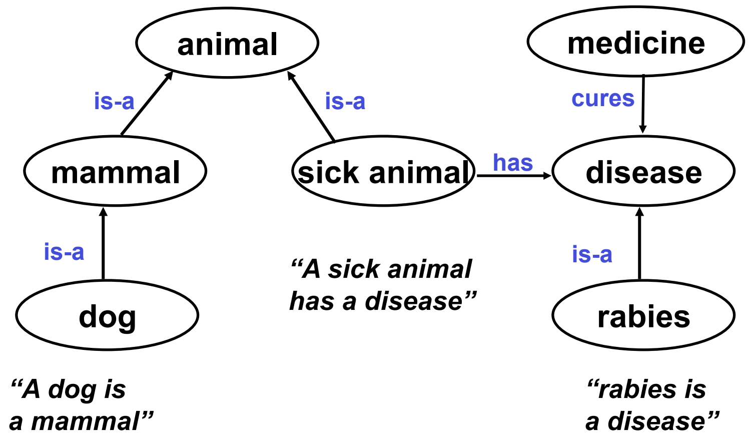
- $\forall x\, (\text{Man}(x) \rightarrow \text{Mortal}(x))$
- $\text{Man}(\text{Socrates})$
- $\therefore \text{Mortal}(\text{Socrates})$

Predicate Calculus (Frege)

Loom KR&R Group

5

Semantic Networks: Nodes and Links



"A sick animal has a disease"

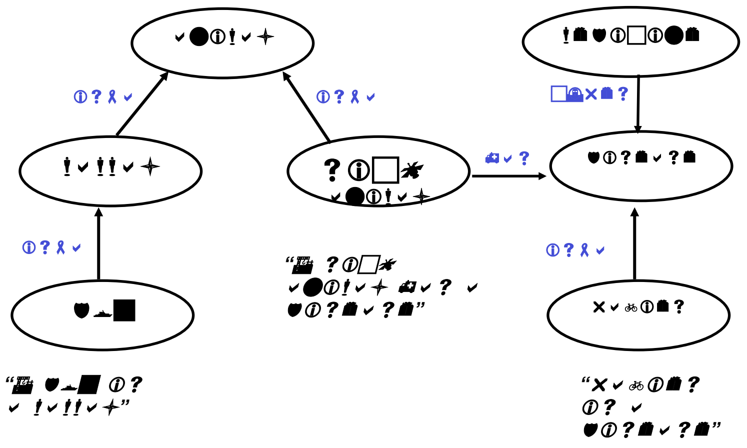
"A dog is a mammal"

"rabies is a disease"

Loom KR&R Group

6

Semantic Networks: The Computer's View

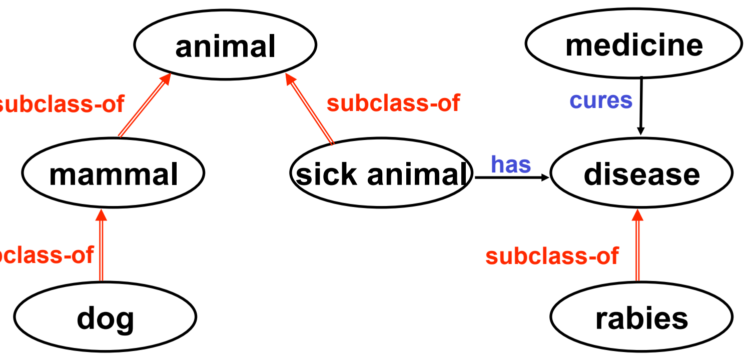


Loom KR&R Group

See: William Woods, “What's in a link: Foundations for Semantic Networks”, 1975

7

Description Logic: Limited Understanding

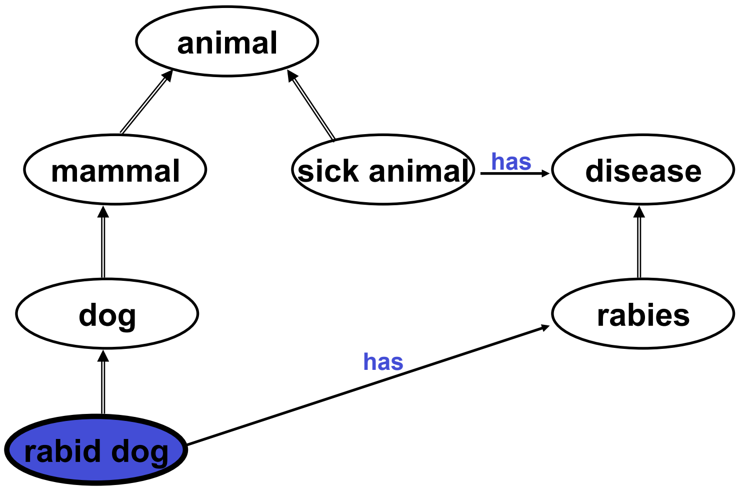


- Subclass relations *“A dog is a mammal”*
- Structural description:
  - Cardinality, Fillers, Type restrictions *“A sick animal has a disease”*

Loom KR&R Group

8

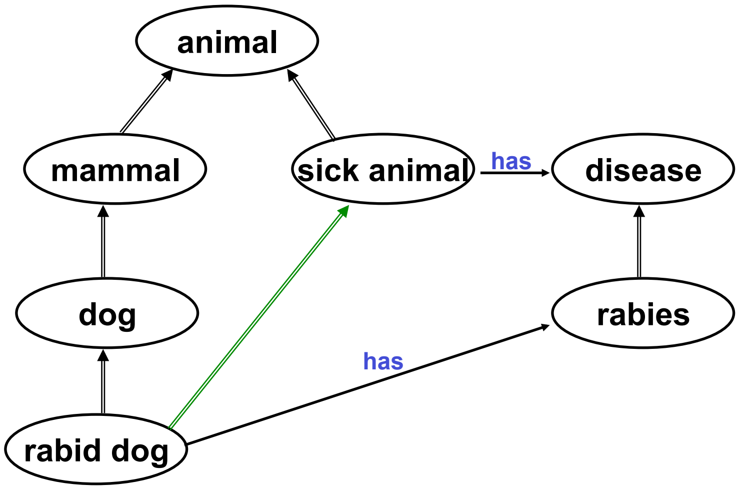
Defining a "rabid dog"



Loom
KR&R
Group

9

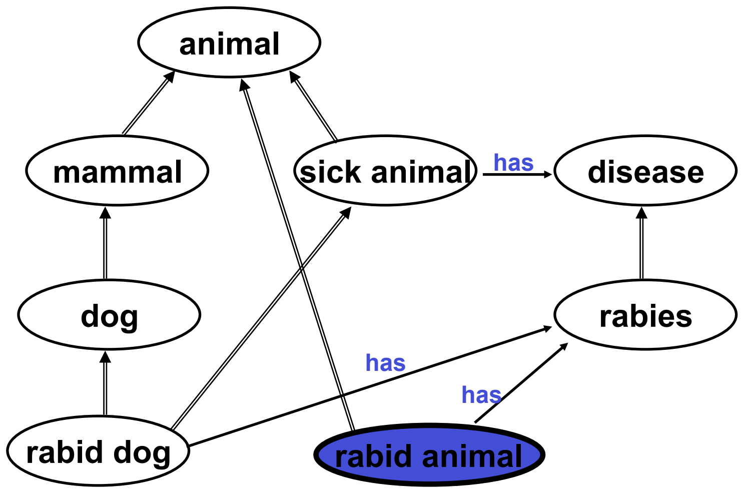
Classification Concludes "sick animal"



Loom
KR&R
Group

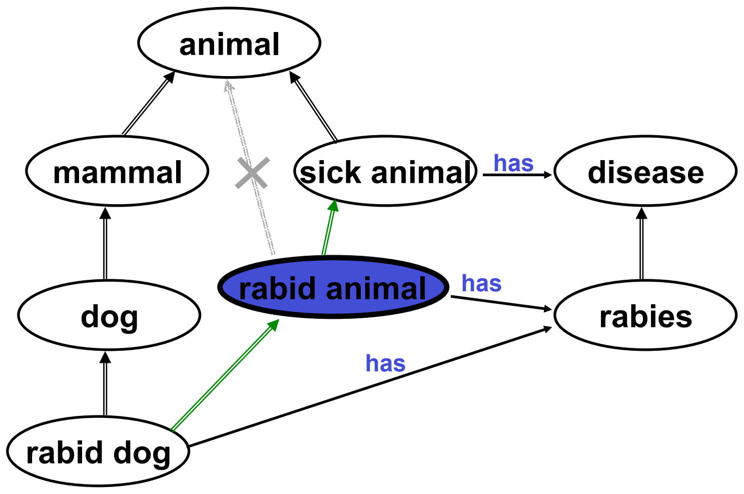
10

Defining “rabid animal”



Loom
KR&R
Group

Classification Places Concept in Hierarchy



Loom
KR&R
Group

Description Logics

- Subsumption is the organizing and reasoning principle
  - Subset-of relation.
- Special language constructs for *structural description*
  - Classifier reasons about subsumption
  - Reasoning is based on *structure* of definitions
  - Limited language to allow tractable inference
    - (all R C)
    - (some R C)
    - (exactly n R)
    - …
- Examples of description logics
  - KL-ONE, KRYPTON, Loom, Classic, OWL

Loom
KR&R
Group

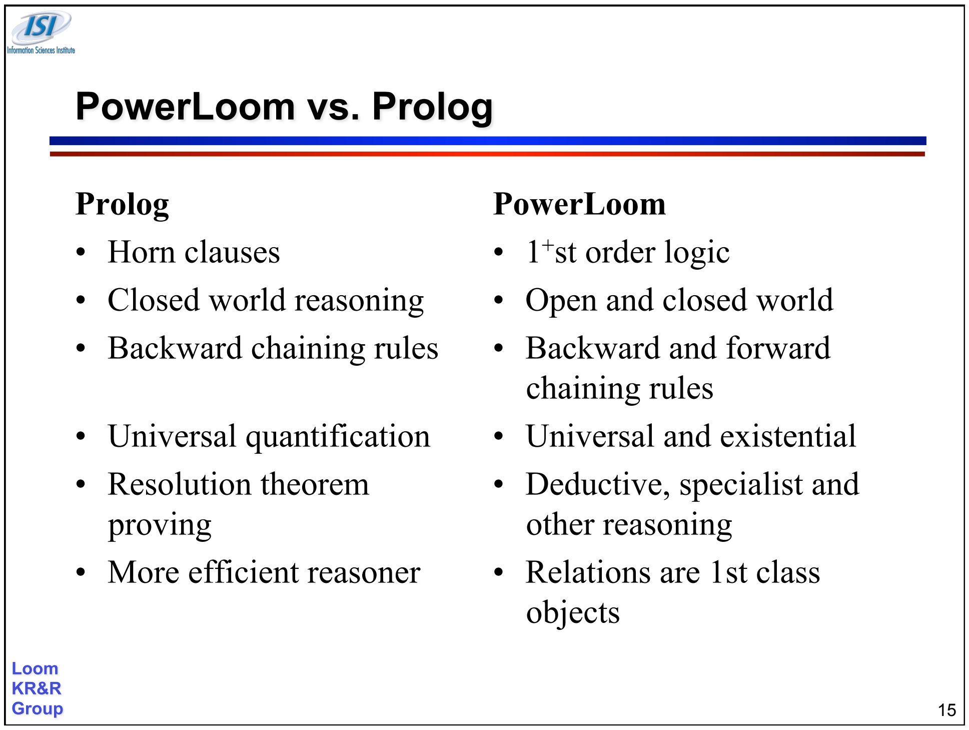
13

Logic and Theorem Provers

- Reasoning based on logic
  - Theorem provers
  - Logic Programming (Prolog)
- PowerLoom combines logical reasoning with ideas from description logics
  - Prolog + additional logical inferences
  - Named concepts and definitions
  - First-order predicate calculus

Loom
KR&R
Group

14

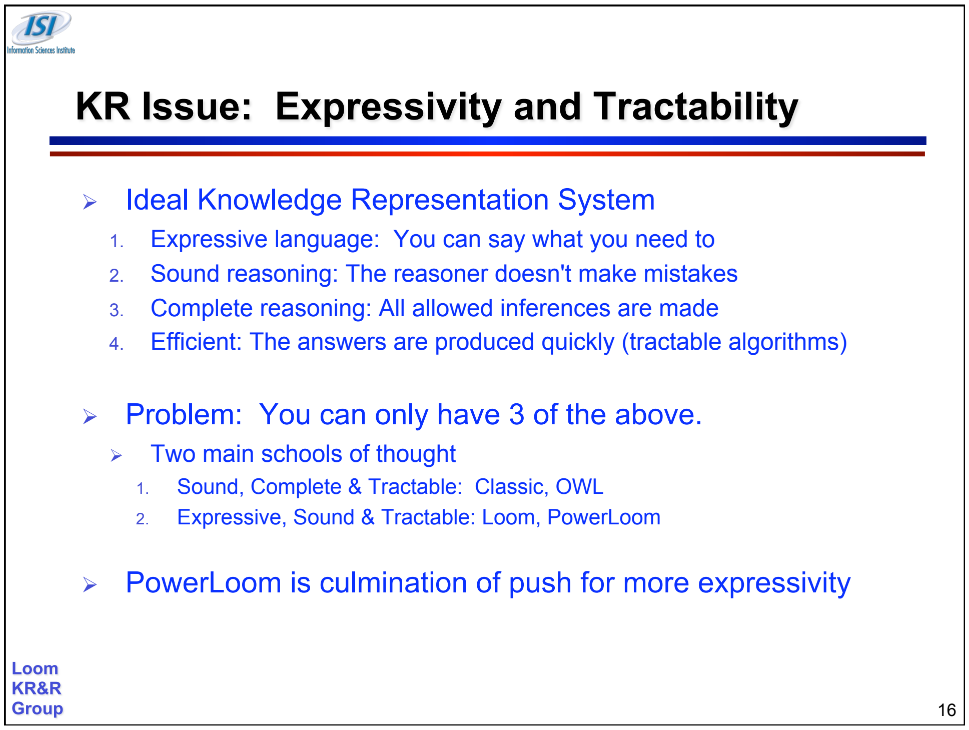


PowerLoom vs. Prolog

Prolog	PowerLoom
• Horn clauses	• 1^{+}st order logic
• Closed world reasoning	• Open and closed world
• Backward chaining rules	• Backward and forward chaining rules
• Universal quantification	• Universal and existential
• Resolution theorem proving	• Deductive, specialist and other reasoning
• More efficient reasoner	• Relations are 1st class objects

Loom
KR&R
Group

15



KR Issue: Expressivity and Tractability

- Ideal Knowledge Representation System
  1. Expressive language: You can say what you need to
  2. Sound reasoning: The reasoner doesn't make mistakes
  3. Complete reasoning: All allowed inferences are made
  4. Efficient: The answers are produced quickly (tractable algorithms)
- Problem: You can only have 3 of the above.
  - Two main schools of thought
    1. Sound, Complete & Tractable: Classic, OWL
    2. Expressive, Sound & Tractable: Loom, PowerLoom
- PowerLoom is culmination of push for more expressivity

Loom
KR&R
Group

16

KR Issue: Closed vs. Open World reasoning

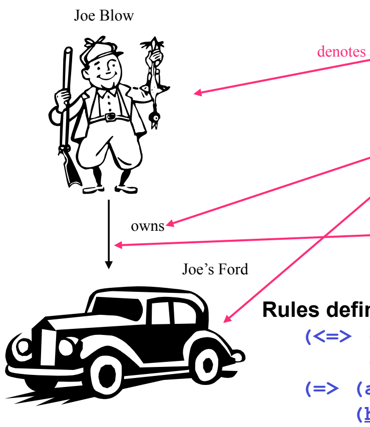
- Closed World means the system knows all relevant facts
  - Allows "negation as failure" reasoning
  - Answers are either true or false
  - Example: President Sample is in this lecture hall — false
  - Database systems and Prolog are closed-world
- Open World means that there may be unknown facts
  - Lack of proof does not mean false
  - Answers are true, false or unknown
  - Example: President Sample is on campus — unknown
  - Many KR systems (including PowerLoom) are open world
    - PowerLoom can also do selectable closed world reasoning

PowerLoom Language

Logical Models 101

"Real" World

Logical Model



Terms represent entities:

`Joe, car002`

Predicates represent relations:

`owns`

Sentences represent what is true in the world (facts):

```
(Person Joe)
(= (age Joe) 17)
(Car car002)
(owns Joe car002)
(model car002 Ford)
(not (rich Joe))
```

Rules define terms and represent domain regularities:

```
(<=> (and (> (age ?x) 12) (< (age ?x) 20))
     (Teenager ?x))
(=> (and (Teenager ?x) (car ?y) (owns ?x ?y))
    (happy ?x))
```

Facts + rules + inference derive concluded facts:

```
(Teenager Joe)
(happy Joe)
```

Loom KR&R Group

19

PowerLoom Representation Language

- First Order Logic base
  - Syntax
  - Declarative semantics
- Prefix notation
- Example:

Facts:
```
(person fred)
(citizen-of fred germany)
(national-language-of germany german)
```

Rules:
```
(forall (?p ?c ?l)
  (=> (and (person ?p)
           (citizen-of ?p ?c)
           (national-language-of ?c ?l))
      (speaks-language ?p ?l)))
```

Loom KR&R Group

20

Definitions

- Terminology (relations, concepts) need to be defined before they are used via **`defconcept`**, **`deffunction`** & **`defrelation`**
- Examples:

```
(defconcept person)
(defrelation married-to ((?p1 person) (?p2 person))
(deffunction + ((?n1 number) (?n2 number))
   :-> (?sum number))
```

- Advantage & Disadvantage
  - Allows certain amount of error checking (e.g., misspelled relations, argument type violations)
  - A bit more tedious and can sometime generate ordering problems

Loom
KR&R
Group

Logical Connectives & Rules

- Predicate logic uses ***logical connectives*** to construct complex sentences from simpler ones:
  - **`and`**, **`or`**, **`not`**, **`<=`**, **`=>`**, **`<=>`**, quantifiers **`exists`** and **`forall`**
- Examples:
  - "Richard is not a crook":
    ```
    (not (crook Richard))
    ```
  - "Every person has a mother":
    ```
    (forall ?p
       (=> (person ?p)
           (exists ?m
             (has-mother ?p ?m))))
    ```

Loom
KR&R
Group

Using PowerLoom

- Starting PowerLoom using Java

```
java -Xmx512m -jar AI.jar
```

or

```
powerloom
```

- Some useful interactive commands
  - Printing or changing modules (contexts)

```
(cc)
(cc "DOG")
```

  - Loading and saving work

```
(load "my-work.plm")
(save-module "DOG" "my-work.plm")
```

  - Getting help

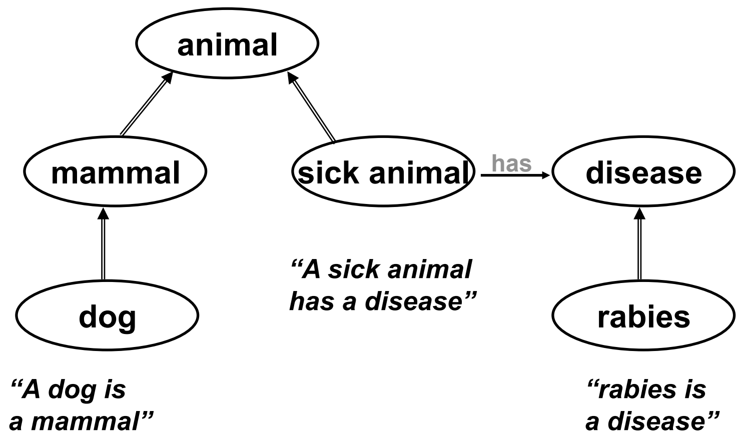
```
(help)
(demo)
```

  - Stopping PowerLoom

```
quit, bye, exit
```

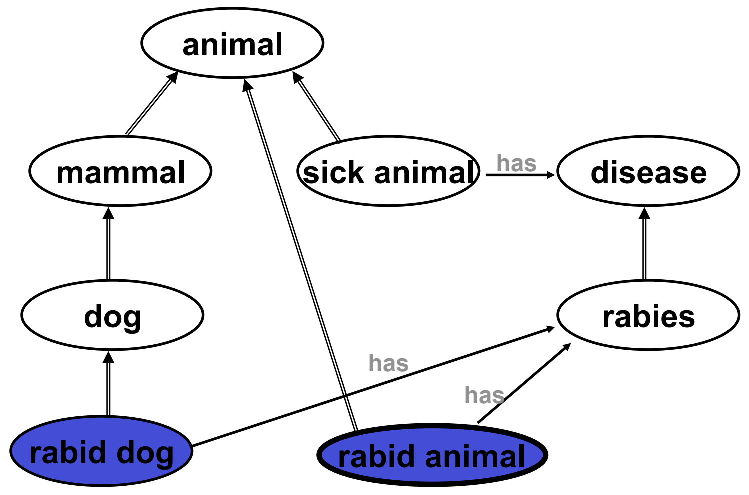
An Example and Demo

Example Domain: Rabies



Loom KR&R Group

Defining "rabid dog" and "rabid animal"



Loom KR&R Group

Quick Demo: Rabid Dog

Loom
KR&R
Group

PowerLoom and Classification

- Classification in PowerLoom is not automatic
- It must be invoked manually
  - `(classify-relations "MY-MODULE" true)`
  - `(classify-instances "MY-MODULE" true)`
- Specific `subset-of` queries will still give the correct answer
  - But value retrieval won't find them
  - Different effort expended – an example of PowerLoom incompleteness.

Loom
KR&R
Group

An Annotated Example

Using Modules

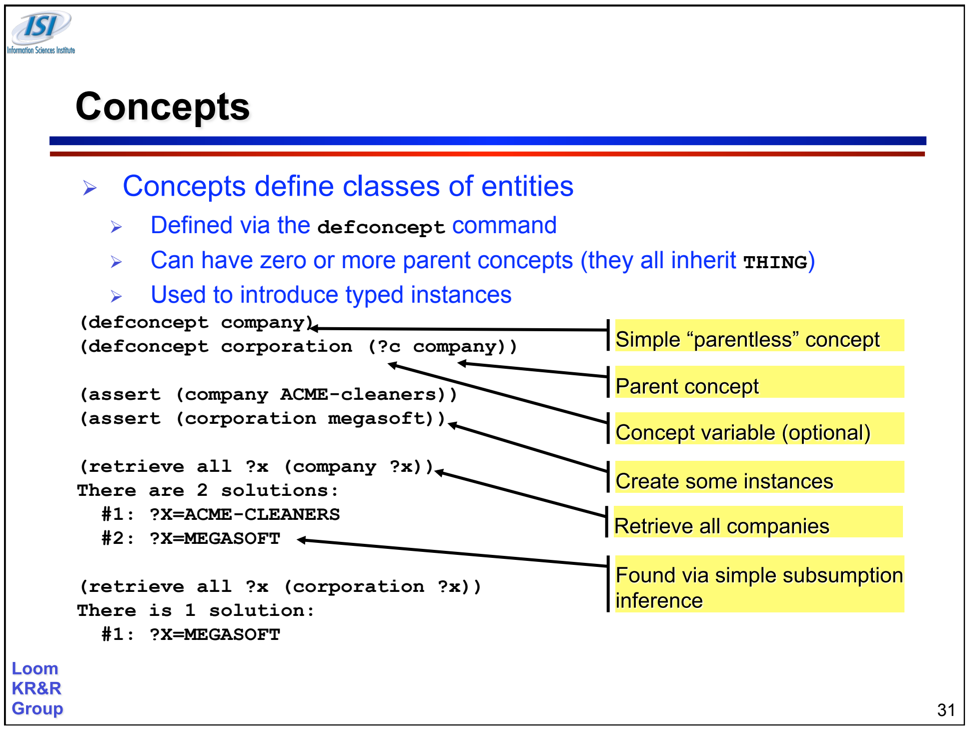
- We define a separate **BUSINESS** module for our example
  - Inherits built-in PowerLoom definitions from **PL-KERNEL/PL-USER**
  - Sets up a separate name and assertion space to avoid unwanted interference with/from other loaded knowledge bases
  - Allows easy experimentation (clearing/changing/editing/saving)
  - All PowerLoom commands are interpreted relative to <u>current module</u>

```
(defmodule "BUSINESS"
  :documentation "Module for the Business demo example."
  :includes ("PL-USER"))
(in-module "BUSINESS")
(clear-module "BUSINESS")
```

List of inherited modules

Set current module

Clear out local content



Concepts

- Concepts define classes of entities
  - Defined via the **`defconcept`** command
  - Can have zero or more parent concepts (they all inherit **`THING`**)
  - Used to introduce typed instances

```
(defconcept company)
(defconcept corporation (?c company))

(assert (company ACME-cleaners))
(assert (corporation megasoft))

(retrieve all ?x (company ?x))
There are 2 solutions:
  #1: ?X=ACME-CLEANERS
  #2: ?X=MEGASOFT

(retrieve all ?x (corporation ?x))
There is 1 solution:
  #1: ?X=MEGASOFT
```

Simple "parentless" concept

Parent concept

Concept variable (optional)

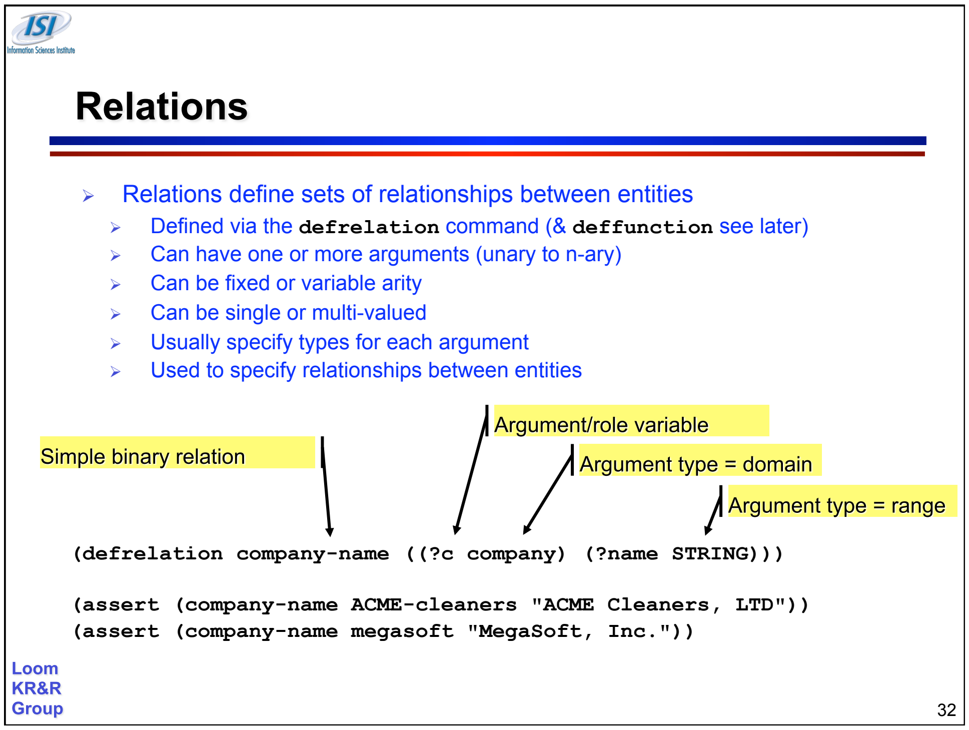
Create some instances

Retrieve all companies

Found via simple subsumption inference

Loom
KR&R
Group

31



Relations

- Relations define sets of relationships between entities
  - Defined via the **`defrelation`** command (& **`deffunction`** see later)
  - Can have one or more arguments (unary to n-ary)
  - Can be fixed or variable arity
  - Can be single or multi-valued
  - Usually specify types for each argument
  - Used to specify relationships between entities

Argument/role variable

Simple binary relation

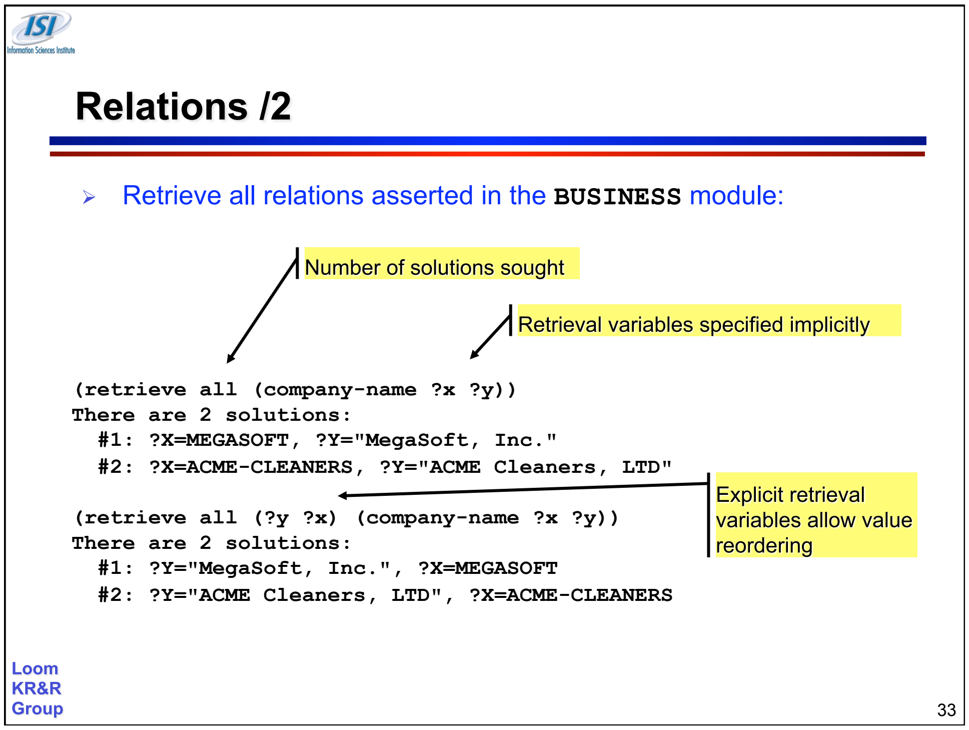
Argument type = domain

Argument type = range

```
(defrelation company-name ((?c company) (?name STRING)))

(assert (company-name ACME-cleaners "ACME Cleaners, LTD"))
(assert (company-name megasoft "MegaSoft, Inc."))
```

Loom
KR&R
Group

32



Relations /2

- Retrieve all relations asserted in the **BUSINESS** module:

Number of solutions sought

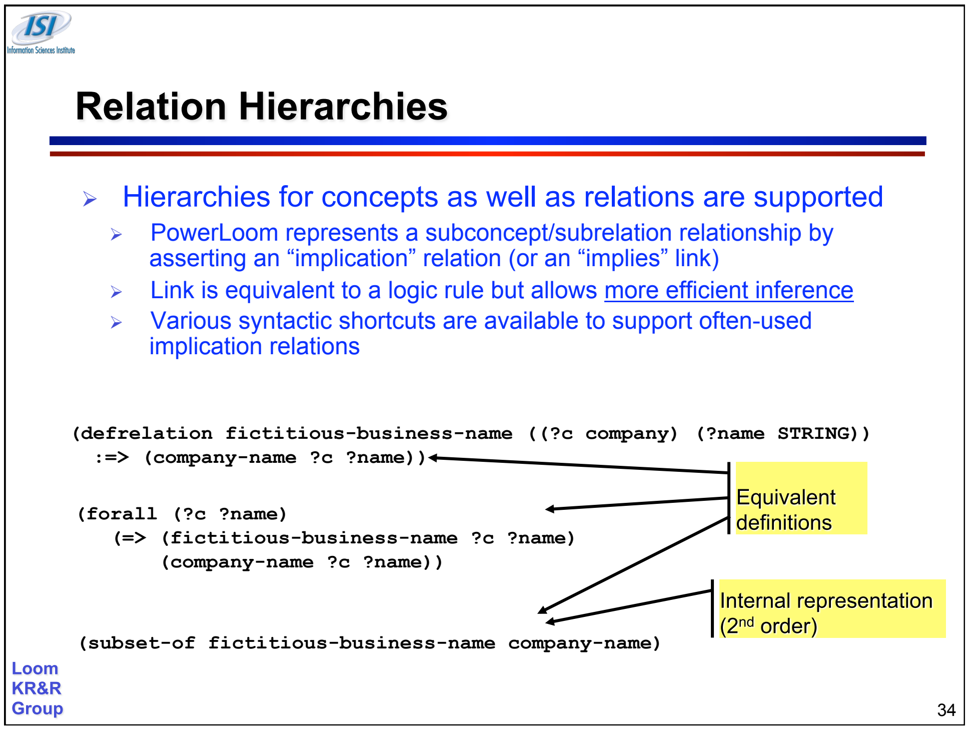
Retrieval variables specified implicitly

```
(retrieve all (company-name ?x ?y))
There are 2 solutions:
  #1: ?X=MEGASOFT, ?Y="MegaSoft, Inc."
  #2: ?X=ACME-CLEANERS, ?Y="ACME Cleaners, LTD"

(retrieve all (?y ?x) (company-name ?x ?y))
There are 2 solutions:
  #1: ?Y="MegaSoft, Inc.", ?X=MEGASOFT
  #2: ?Y="ACME Cleaners, LTD", ?X=ACME-CLEANERS
```

Explicit retrieval variables allow value reordering

Loom
KR&R
Group

33



Relation Hierarchies

- Hierarchies for concepts as well as relations are supported
  - PowerLoom represents a subconcept/subrelation relationship by asserting an “implication” relation (or an “implies” link)
  - Link is equivalent to a logic rule but allows <u>more efficient inference</u>
  - Various syntactic shortcuts are available to support often-used implication relations

```
(defrelation fictitious-business-name ((?c company) (?name STRING))
  :=> (company-name ?c ?name))

(forall (?c ?name)
   (=> (fictitious-business-name ?c ?name)
       (company-name ?c ?name))

(subset-of fictitious-business-name company-name)
```

Equivalent definitions

Internal representation (2nd order)

Loom
KR&R
Group

34

Relation Hierarchies /2

- Retrieve all names of MegaSoft, fictitious or not
  - Illustrates that **`company-name`** is a multi-valued relation

```
(assert (fictitious-business-name megasoft "MegaSoft"))


(retrieve all ?x (company-name megasoft ?x))
There are 2 solutions:
  #1: ?X="MegaSoft, Inc."
  #2: ?X="MegaSoft"
```

Directly asserted

Inferred via the subrelation rule/link

Functions

- Functions are term-producing, single-valued relations
  - Defined via the **`deffunction`** command
  - Very similar to relations defined via **`defrelation`** but:
  - Term producing: a function applied to its first n-1 input arguments specifies a unique, intensional term, e.g., “Fred’s age”
  - Single-valued: each set of input arguments has at most one output argument (the last argument), e.g., “Fred’s age is 42”
  - By default, functions are assumed to be partial, i.e., could be undefined for some legal input values (e.g., 1/0)

Input argument

Output argument

```
(deffunction number-of-employees ((?c company)) :-> (?n INTEGER))
```

Function term

Function value

```
(assert (= (number-of-employees ACME-cleaners) 8))
(assert (= (number-of-employees megasoft) 10000))
```

Functions /2

- Functions syntax often results in shorter expressions than using similar relation syntax:

```
(retrieve all (and (company ?x)
                   (< (number-of-employees ?x) 50)))
There is 1 solution:
  #1: ?X=ACME-CLEANERS
```

- Compare to:

```
(retrieve all (and (company ?x)
                   (exists ?n
                     (and (number-of-employees ?x ?n)
                          (< ?n 50)))))
There is 1 solution:
  #1: ?X=ACME-CLEANERS
```

- Multiple function terms:

```
(retrieve all (> (number-of-employees ?x) (number-of-employees ?y)))
There is 1 solution:
  #1: ?X=MEGASOFT, ?Y=ACME-CLEANERS
```

Loom
KR&R
Group

37

Defined Concepts

- Concepts (and functions/relations) can be defined completely in terms of rules
  - Useful to name often-used queries or subexpressions and build up powerful vocabulary

```
(defconcept small-company (?c company)
  :<=> (and (company ?c)
            (< (number-of-employees ?c) 50)))
```

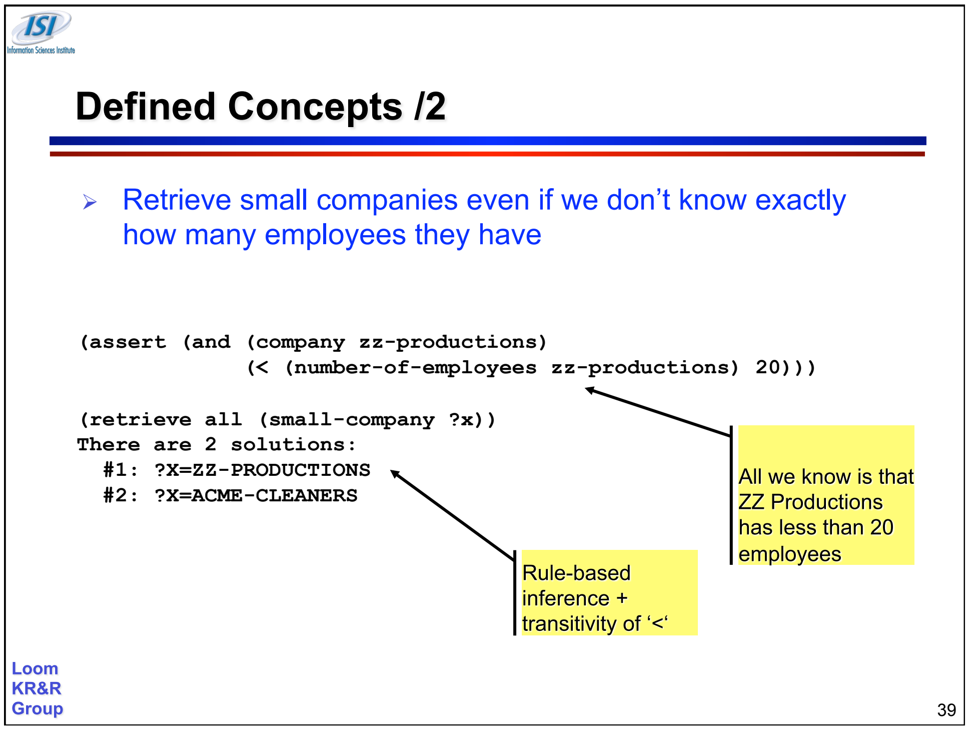
New keyword

Expands into these rules

```
(forall ?c (=> (and (company ?c)
                    (< (number-of-employees ?c) 50))
               (small-company ?c)))

(forall ?c (=> (small-company ?c)
               (and (company ?c)
                    (< (number-of-employees ?c) 50))))
```

Loom
KR&R
Group

38



Defined Concepts /2

- Retrieve small companies even if we don’t know exactly how many employees they have

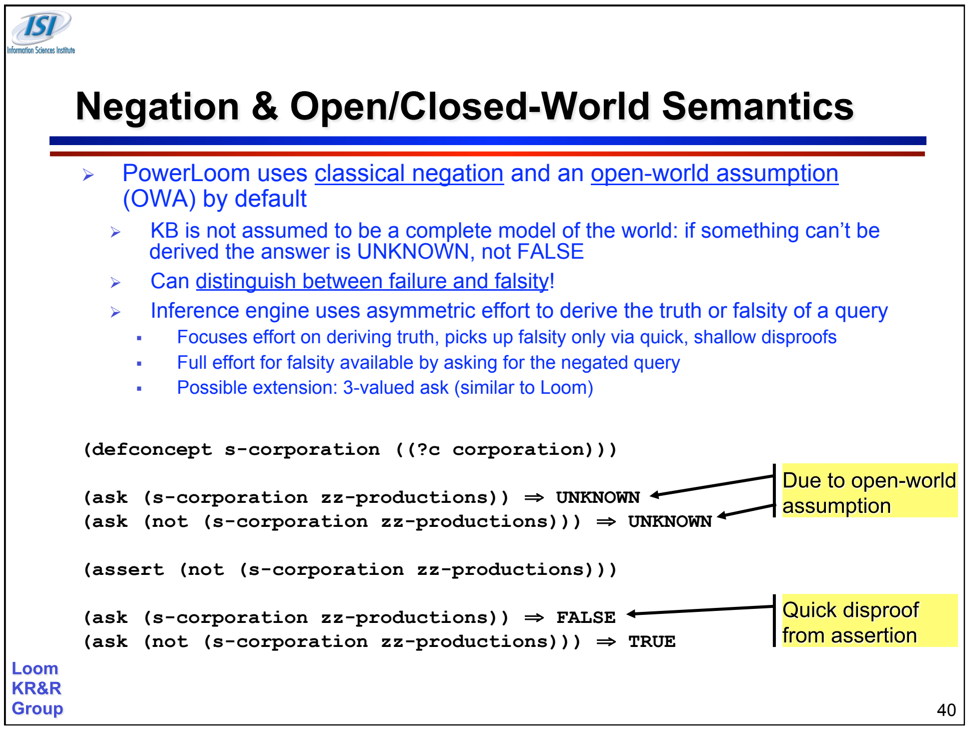
```
(assert (and (company zz-productions)
             (< (number-of-employees zz-productions) 20)))

(retrieve all (small-company ?x))
There are 2 solutions:
  #1: ?X=ZZ-PRODUCTIONS
  #2: ?X=ACME-CLEANERS
```

All we know is that ZZ Productions has less than 20 employees

Rule-based inference + transitivity of ‘<‘

Loom KR&R Group

39



Negation & Open/Closed-World Semantics

- PowerLoom uses classical negation and an open-world assumption (OWA) by default
  - KB is not assumed to be a complete model of the world: if something can’t be derived the answer is UNKNOWN, not FALSE
  - Can distinguish between failure and falsity!
  - Inference engine uses asymmetric effort to derive the truth or falsity of a query
    - Focuses effort on deriving truth, picks up falsity only via quick, shallow disproofs
    - Full effort for falsity available by asking for the negated query
    - Possible extension: 3-valued ask (similar to Loom)

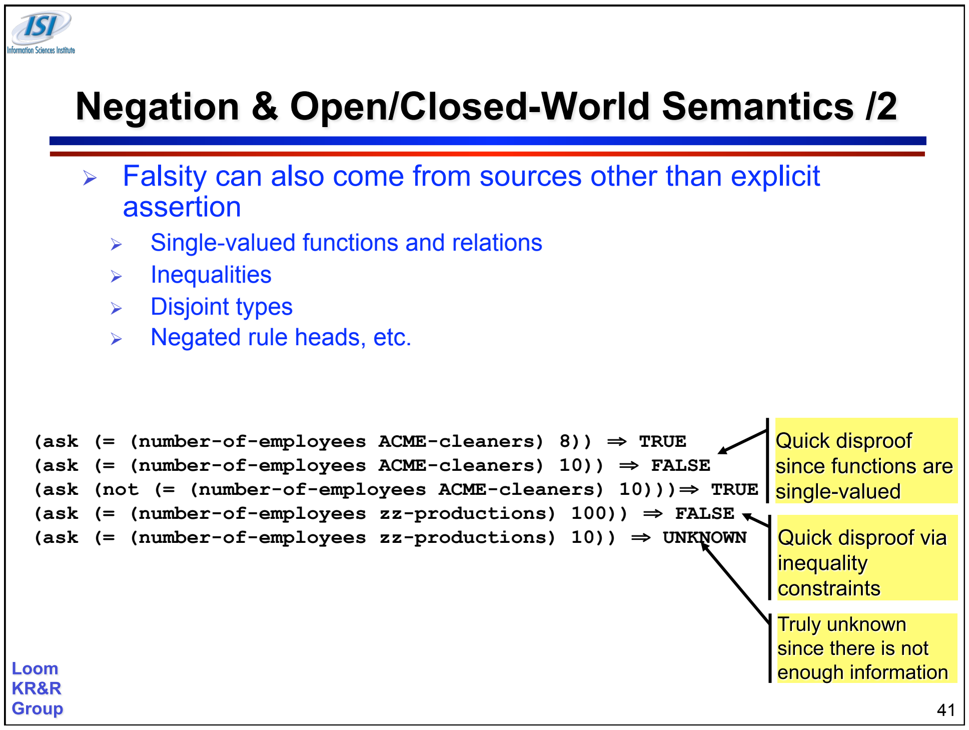
```
(defconcept s-corporation ((?c corporation)))

(ask (s-corporation zz-productions)) ⇒ UNKNOWN
(ask (not (s-corporation zz-productions))) ⇒ UNKNOWN

(assert (not (s-corporation zz-productions)))

(ask (s-corporation zz-productions)) ⇒ FALSE
(ask (not (s-corporation zz-productions))) ⇒ TRUE
```

Due to open-world assumption

Quick disproof from assertion

Loom KR&R Group

40



Negation & Open/Closed-World Semantics /2

- Falsity can also come from sources other than explicit assertion
  - Single-valued functions and relations
  - Inequalities
  - Disjoint types
  - Negated rule heads, etc.

```
(ask (= (number-of-employees ACME-cleaners) 8)) ⇒ TRUE
(ask (= (number-of-employees ACME-cleaners) 10)) ⇒ FALSE
(ask (not (= (number-of-employees ACME-cleaners) 10)))⇒ TRUE
(ask (= (number-of-employees zz-productions) 100)) ⇒ FALSE
(ask (= (number-of-employees zz-productions) 10)) ⇒ UNKNOWN
```

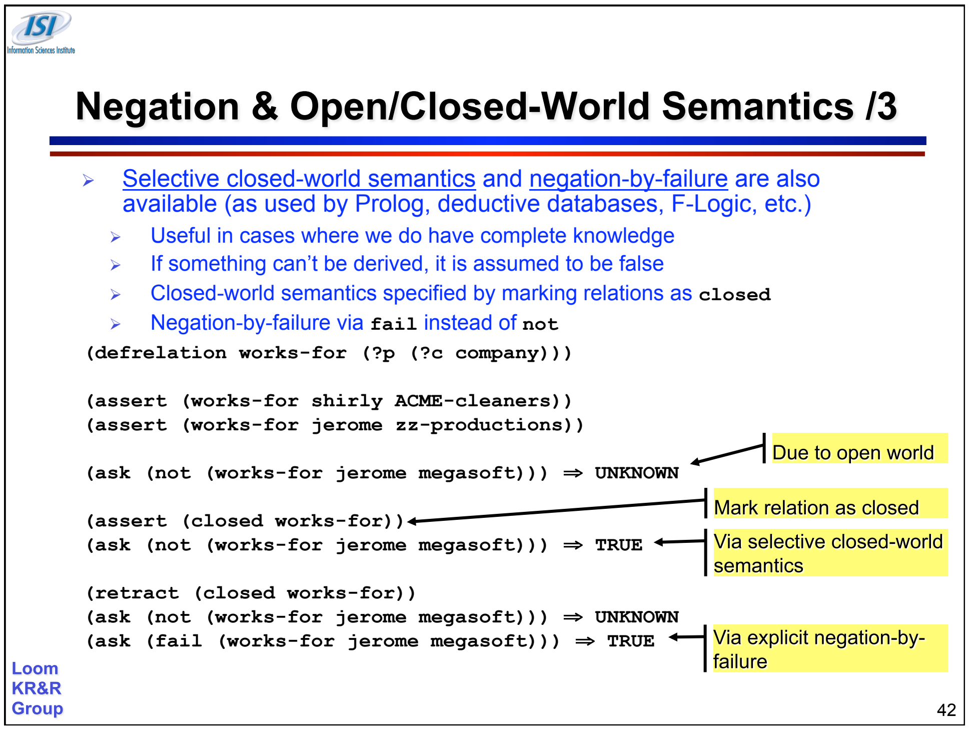
Quick disproof since functions are single-valued

Quick disproof via inequality constraints

Truly unknown since there is not enough information

Loom
KR&R
Group

41



Negation & Open/Closed-World Semantics /3

- Selective closed-world semantics and negation-by-failure are also available (as used by Prolog, deductive databases, F-Logic, etc.)
  - Useful in cases where we do have complete knowledge
  - If something can't be derived, it is assumed to be false
  - Closed-world semantics specified by marking relations as `closed`
  - Negation-by-failure via `fail` instead of `not`

```
(defrelation works-for (?p (?c company)))

(assert (works-for shirly ACME-cleaners))
(assert (works-for jerome zz-productions))

(ask (not (works-for jerome megasoft))) ⇒ UNKNOWN

(assert (closed works-for))
(ask (not (works-for jerome megasoft))) ⇒ TRUE

(retract (closed works-for))
(ask (not (works-for jerome megasoft))) ⇒ UNKNOWN
(ask (fail (works-for jerome megasoft))) ⇒ TRUE
```

Due to open world

Mark relation as closed

Via selective closed-world semantics

Via explicit negation-by-failure

Loom
KR&R
Group

42

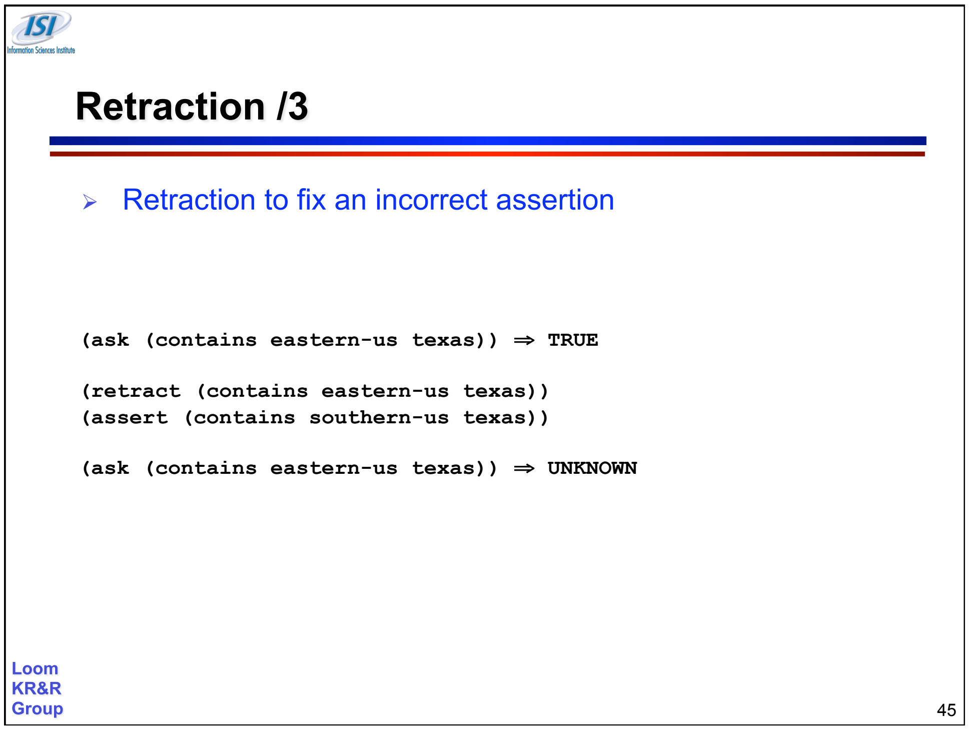
Retraction

- Retraction allows the erasure or change of a previously asserted truth-value of a proposition
  - Useful for error correction or iterative “change of mind” during development
  - Useful to change certain aspects of a scenario without having to reload the whole knowledge base
  - Allows efficient, fine-grained change
    - Some cached information is lost and needs to be regenerated
    - Loss can be minimized by careful structuring of module hierarchy (put more stable knowledge higher up in the hierarchy)
  - Allows the exploration of hypothetical conjectures
    - What would change if F were true or false?
    - Module system allows us to consider both possibilities at the same time

Retraction /2

- Some geographic terminology and information

```
(defconcept geographic-location)
(defconcept country ((?l geographic-location)))
(defconcept state ((?l geographic-location)))
(defconcept city ((?l geographic-location)))
(defrelation contains ((?l1 geographic-location)
                       (?l2 geographic-location)))

(assert (and
         (country united-states)
         (geographic-location eastern-us)
         (contains united-states eastern-us)
         (state georgia) (contains eastern-us georgia)
         (city atlanta) (contains georgia atlanta)
         (geographic-location southern-us)
         (contains united-states southern-us)
         (state texas) (contains eastern-us texas)
         (city dallas) (contains texas dallas)
         (city austin) (contains texas austin)))
```



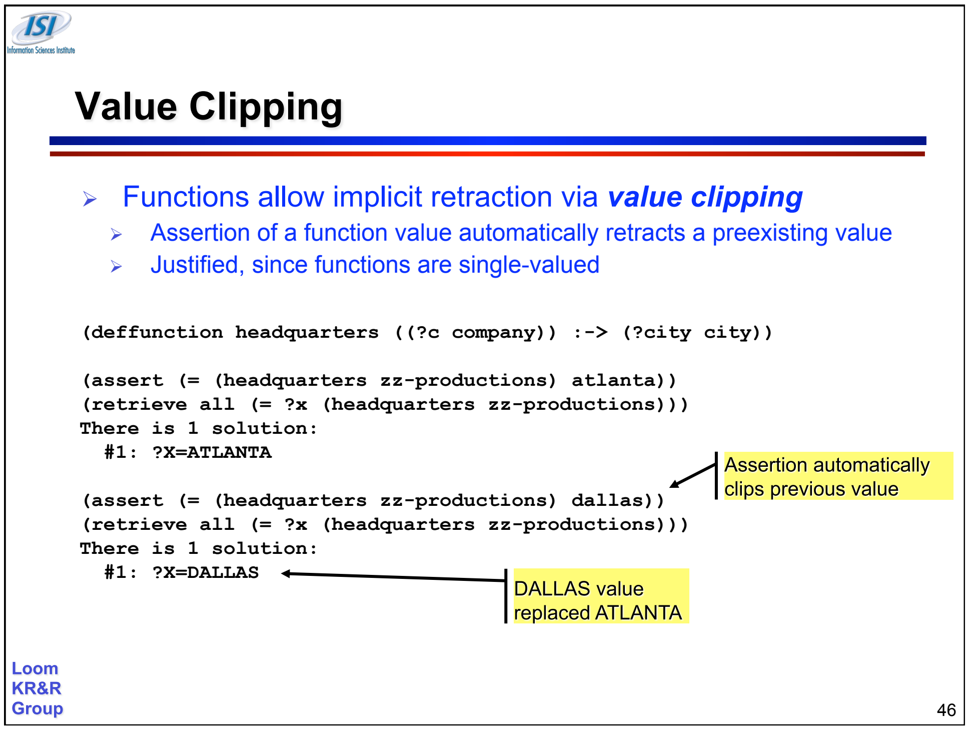
Retraction /3

- Retraction to fix an incorrect assertion

```
(ask (contains eastern-us texas)) ⇒ TRUE

(retract (contains eastern-us texas))
(assert (contains southern-us texas))

(ask (contains eastern-us texas)) ⇒ UNKNOWN
```

Loom
KR&R
Group

45



Value Clipping

- Functions allow implicit retraction via ***value clipping***
  - Assertion of a function value automatically retracts a preexisting value
  - Justified, since functions are single-valued

```
(deffunction headquarters ((?c company)) :-> (?city city))

(assert (= (headquarters zz-productions) atlanta))
(retrieve all (= ?x (headquarters zz-productions)))
There is 1 solution:
  #1: ?X=ATLANTA

(assert (= (headquarters zz-productions) dallas))
(retrieve all (= ?x (headquarters zz-productions)))
There is 1 solution:
  #1: ?X=DALLAS
```

Assertion automatically clips previous value

DALLAS value replaced ATLANTA

Loom
KR&R
Group

46

Value Clipping /2

- Clipping also works for single-valued relations

```
(defrelation headquartered-in ((?c company) (?city city))
  :axioms (single-valued headquartered-in))

(assert (headquartered-in megasoft atlanta))
(retrieve all (headquartered-in megasoft ?x))
There is 1 solution:
  #1: ?X=ATLANTA

(assert (headquartered-in megasoft dallas))
(retrieve all (headquartered-in megasoft ?x))
There is 1 solution:
  #1: ?X=DALLAS
```

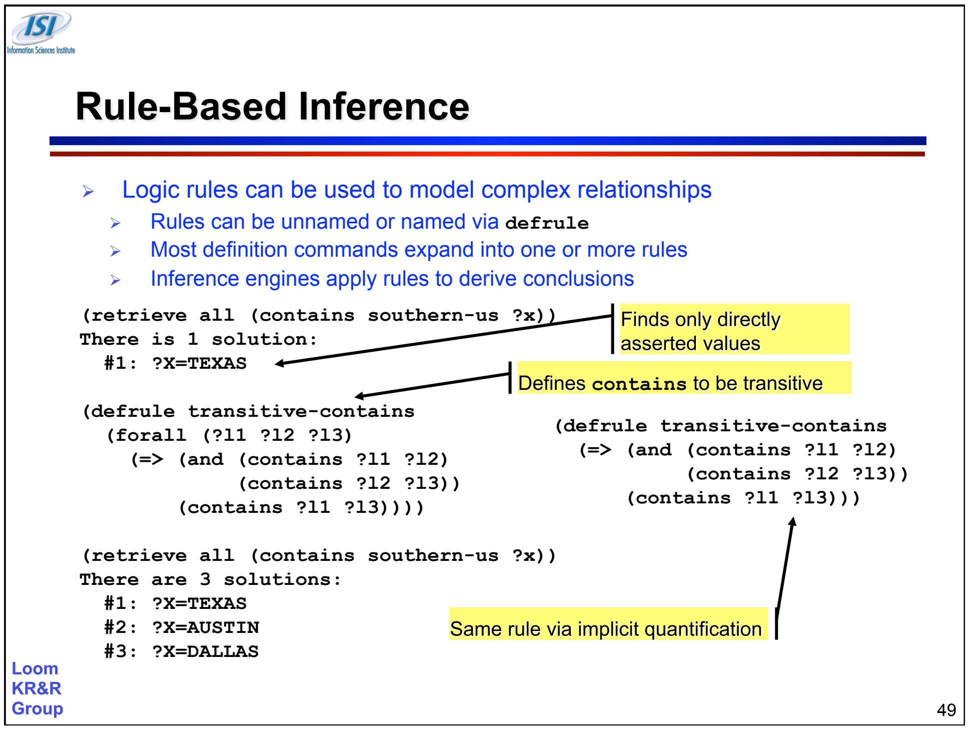
Loom
KR&R
Group

Contradictions

- Propositions that are both `TRUE` and `FALSE` are contradictory
  - Contradictions can result from explicit assertions, during forward-chaining, or as the result of a refutation proof
  - Contradictory propositions are treated as `UNKNOWN` to allow the system to continue to function

```
(assert (not (state texas)))

Derived both TRUE and FALSE for the proposition `|P#|(STATE TEXAS)'.
   Clash occurred in module `|MDL|/PL-KERNEL-KB/BUSINESS'.

(ask (state texas)) ⇒ UNKNOWN
(ask (not (state texas))) ⇒ UNKNOWN
```

Loom
KR&R
Group



Rule-Based Inference

- Logic rules can be used to model complex relationships
  - Rules can be unnamed or named via `defrule`
  - Most definition commands expand into one or more rules
  - Inference engines apply rules to derive conclusions

```
(retrieve all (contains southern-us ?x))
There is 1 solution:
  #1: ?X=TEXAS
```

Finds only directly asserted values

Defines `contains` to be transitive

```
(defrule transitive-contains
  (forall (?l1 ?l2 ?l3)
    (=> (and (contains ?l1 ?l2)
             (contains ?l2 ?l3))
        (contains ?l1 ?l3))))
```

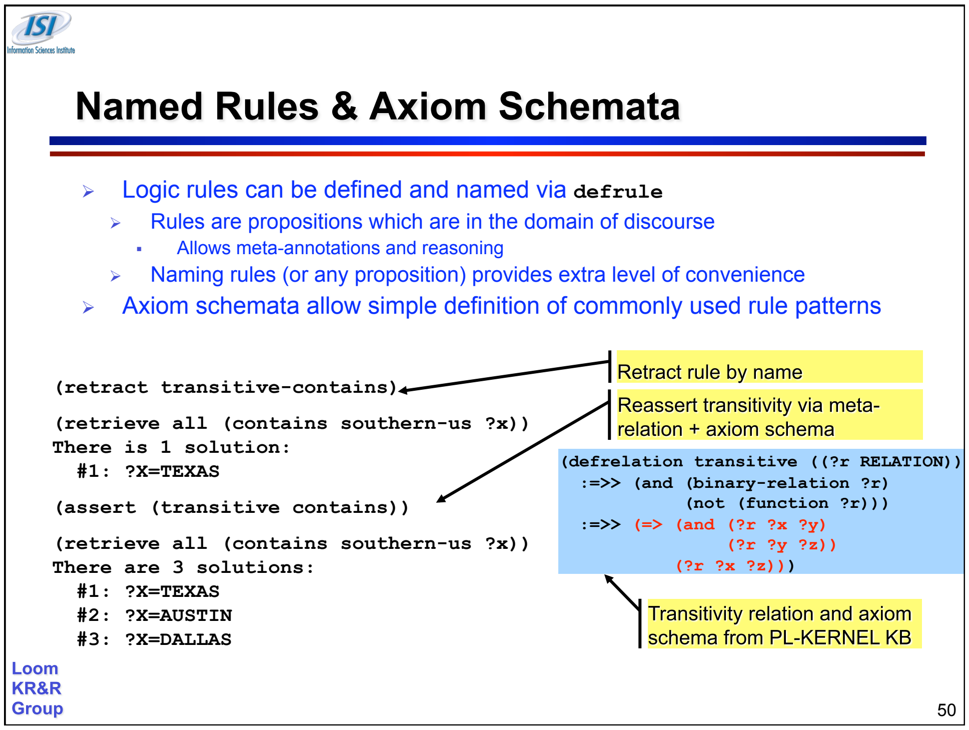
```
(defrule transitive-contains
  (=> (and (contains ?l1 ?l2)
           (contains ?l2 ?l3))
      (contains ?l1 ?l3)))
```

```
(retrieve all (contains southern-us ?x))
There are 3 solutions:
  #1: ?X=TEXAS
  #2: ?X=AUSTIN
  #3: ?X=DALLAS
```

Same rule via implicit quantification

Loom
KR&R
Group

49



Named Rules & Axiom Schemata

- Logic rules can be defined and named via `defrule`
  - Rules are propositions which are in the domain of discourse
    - Allows meta-annotations and reasoning
  - Naming rules (or any proposition) provides extra level of convenience
- Axiom schemata allow simple definition of commonly used rule patterns

```
(retract transitive-contains)

(retrieve all (contains southern-us ?x))
There is 1 solution:
  #1: ?X=TEXAS

(assert (transitive contains))

(retrieve all (contains southern-us ?x))
There are 3 solutions:
  #1: ?X=TEXAS
  #2: ?X=AUSTIN
  #3: ?X=DALLAS
```

Retract rule by name

Reassert transitivity via meta-relation + axiom schema

```
(defrelation transitive ((?r RELATION))
  :=>> (and (binary-relation ?r)
            (not (function ?r)))
  :=>> (=> (and (?r ?x ?y)
                (?r ?y ?z))
           (?r ?x ?z)))
```

Transitivity relation and axiom schema from PL-KERNEL KB

Loom
KR&R
Group

50

Justifications and Explanation

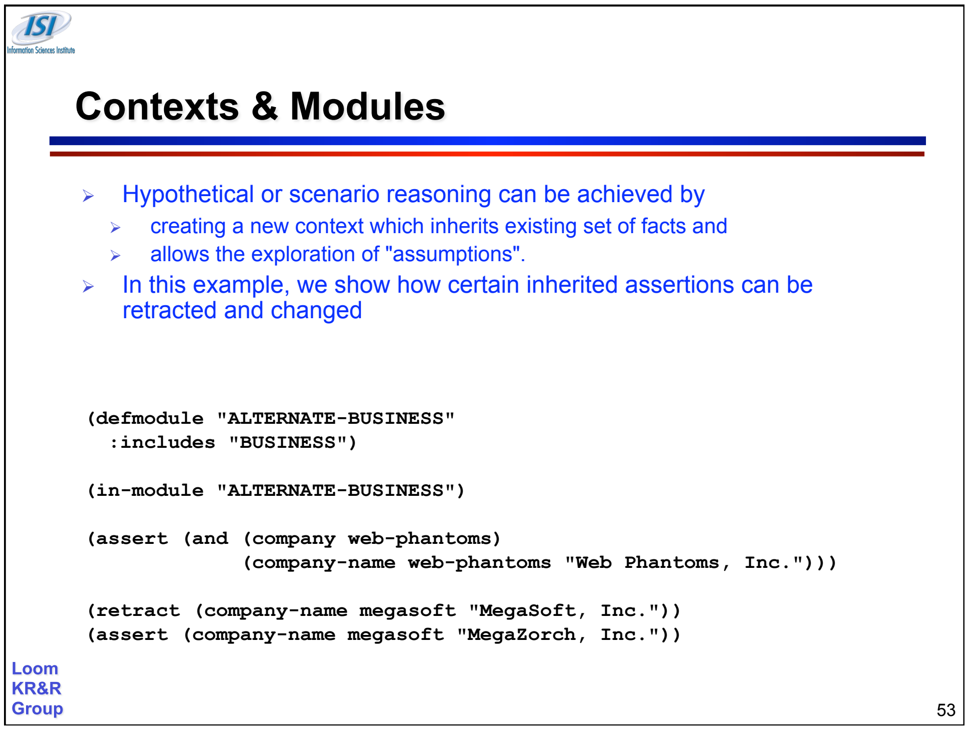
- Explanation of true/false queries
  - Backward inference can store proof trees that can be rendered into explanations
  - Simple built-in explanation mechanism
    - Various rendering possibilities, ASCII, HTML, XML
    - Eliminates explanation of duplicate and low-level goals
    - Explanation strings for different audiences (technical, lay)

```
(ask (contains southern-us dallas)) ⇒ TRUE

(why)
1 (CONTAINS SOUTHERN-US DALLAS)
    follows by Modus Ponens
    with substitution {?l1/SOUTHERN-US, ?l3/DALLAS, ?l2/TEXAS}
    since 1.1 ! (FORALL (?l1 ?l3)
                   (<= (CONTAINS ?l1 ?l3)
                       (EXISTS (?l2)
                          (AND (CONTAINS ?l1 ?l2)
                               (CONTAINS ?l2 ?l3)))))
    and   1.2 ! (CONTAINS SOUTHERN-US TEXAS)
    and   1.3 ! (CONTAINS TEXAS DALLAS)
```

Explanation /2

- Explanation of retrieved results
  - Separate explanation for each derived solution
  - `why` explains most recently retrieved solution

```
(retrieve 3 (contains southern-us ?x))
There are 3 solutions so far:
  #1: ?X=DALLAS
  #2: ?X=TEXAS
  #3: ?X=AUSTIN

(why)
1 (CONTAINS SOUTHERN-US AUSTIN)
    follows by Modus Ponens
    with substitution {?l1/SOUTHERN-US, ?l3/AUSTIN, ?l2/TEXAS}
    since 1.1 ! (FORALL (?l1 ?l3)
                   (<= (CONTAINS ?l1 ?l3)
                       (EXISTS (?l2)
                          (AND (CONTAINS ?l1 ?l2)
                               (CONTAINS ?l2 ?l3)))))
    and   1.2 ! (CONTAINS SOUTHERN-US TEXAS)
    and   1.3 ! (CONTAINS TEXAS AUSTIN)
```



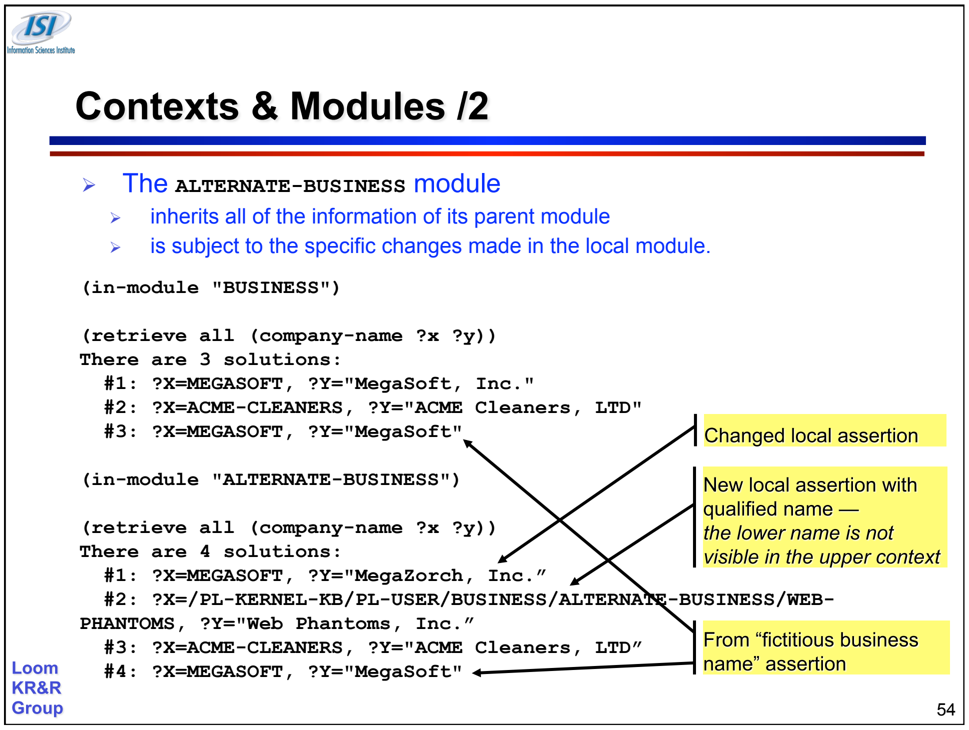
Contexts & Modules

- Hypothetical or scenario reasoning can be achieved by
  - creating a new context which inherits existing set of facts and
  - allows the exploration of "assumptions".
- In this example, we show how certain inherited assertions can be retracted and changed

```
(defmodule "ALTERNATE-BUSINESS"
  :includes "BUSINESS")

(in-module "ALTERNATE-BUSINESS")

(assert (and (company web-phantoms)
             (company-name web-phantoms "Web Phantoms, Inc.")))

(retract (company-name megasoft "MegaSoft, Inc."))
(assert (company-name megasoft "MegaZorch, Inc."))
```

Loom
KR&R
Group

53



Contexts & Modules /2

- The `ALTERNATE-BUSINESS` module
  - inherits all of the information of its parent module
  - is subject to the specific changes made in the local module.

```
(in-module "BUSINESS")

(retrieve all (company-name ?x ?y))
There are 3 solutions:
  #1: ?X=MEGASOFT, ?Y="MegaSoft, Inc."
  #2: ?X=ACME-CLEANERS, ?Y="ACME Cleaners, LTD"
  #3: ?X=MEGASOFT, ?Y="MegaSoft"

(in-module "ALTERNATE-BUSINESS")

(retrieve all (company-name ?x ?y))
There are 4 solutions:
  #1: ?X=MEGASOFT, ?Y="MegaZorch, Inc."
  #2: ?X=/PL-KERNEL-KB/PL-USER/BUSINESS/ALTERNATE-BUSINESS/WEB-
PHANTOMS, ?Y="Web Phantoms, Inc."
  #3: ?X=ACME-CLEANERS, ?Y="ACME Cleaners, LTD"
  #4: ?X=MEGASOFT, ?Y="MegaSoft"
```

Changed local assertion

New local assertion with qualified name — *the lower name is not visible in the upper context*

From “fictitious business name” assertion

Loom
KR&R
Group

54

Cross-Contextual Reasoning

- Normally queries operate in the current module.
  - The IST (IS-TRUE) relation (J. McCarthy) allows us to query about the state of knowledge in other modules.
  - This also allows cross-module inference by binding variables across forms
  - Example: “find all companies whose names differ in the two modules”

```
(in-module "BUSINESS")

(retrieve all (ist alternate-business (company-name ?x ?y)))
There are 4 solutions:
  #1: ?X=MEGASOFT, ?Y="MegaZorch, Inc."
  #2: ?X=MEGASOFT, ?Y="MegaSoft, Inc."
  #3: ?X=ACME-CLEANERS, ?Y="ACME Cleaners, LTD"
  #4: ?X=MEGASOFT, ?Y="MegaSoft"

(retrieve all (and (ist business (company-name ?x ?y))
                   (fail (ist alternate-business (company-name ?x ?y)))))
There is 1 solution:
  #1: ?X=MEGASOFT, ?Y="MegaSoft, Inc."
```

Using PowerLoom from Java

Java Setup

- Details in the PowerLoom Manual
- Mapping PowerLoom names
  - Follows standard Java conventions
    - `s-assert-proposition` ⇒ `sAssertProposition`
  - "*" character maps to "$"
    - `*module*` ⇒ `$MODULE$` — it's a global variable!
  - "?" character maps to "P" (for Predicate)
    - `next?` ⇒ `nextP`
- Java import statements

```
import edu.isi.powerloom.*;
import edu.isi.powerloom.logic.*;
import edu.isi.stella.Module;
import edu.isi.stella.Stella_Object;
```

Initialization and Loading Files

- PowerLoom needs to be initialized before using. This can take a while. This form initializes basic PowerLoom
  - `PLI.initialize();`
- Other systems may also need initialization.
  - For example, PowerLoom extensions to get units and dimensions:
    - `StartupPowerloomSystem.startupPowerloomSystem();`
- PowerLoom files may need loading
  - `PLI.load("mykb.plm", null);`

Assertions, Retractions and Definitions

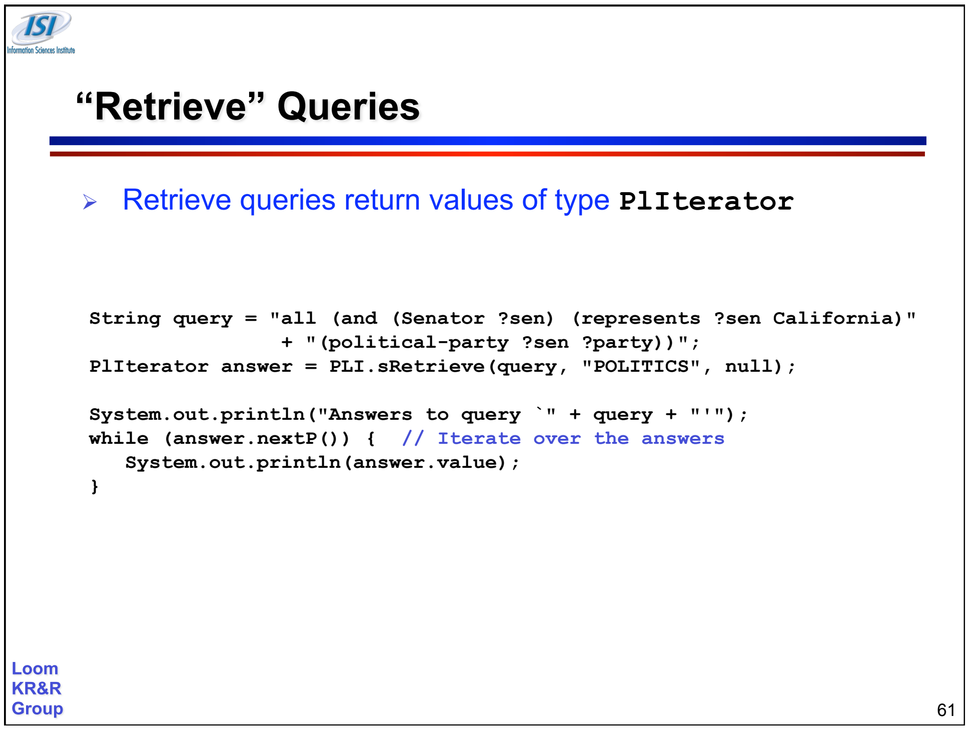
- Almost all needed interface methods are in the PLI class as static methods.
- Many have both object and String interfaces. Strings are generally easier to use.
- The general `sEvaluate` form can process any command that can be given at the interactive prompt.
- Most methods take a module and environment argument. The environment can be left as `null` to use the default.

```
PLI.sAssertProposition("(Person Fred)", "PL-USER", null);
PLI.sAssertProposition("(name Fred \"Frederick\")", "PL-USER", null);

PLI.sRetractProposition("(Hungry Fred)", "PL-USER", null);

PLI.sCreateRelation("friend", 2, "PL-USER", null);
PLI.sEvaluate("(deffunction age ((?p Person) (?n INTEGER))",
              "PL-USER", null);
```

"Ask" Queries

- Ask queries return values of type `TruthValue`
- `PLI` has predicates to test the returned values.

```
PLI.isTrue(PLI.sAsk("(> 8 7)", "PL-USER", null));


TruthValue tv = PLI.sAsk("(friend Jobs Eisner)", "PL-USER", null);

if (isTrue(tv))    System.out.println("Yes!");
if (isFalse(tv))   System.out.println("No.");
if (isUnknown(tv)) System.out.println("How should I know?");
if (isDefault(tv)) System.out.println("  by default reasoning");
```



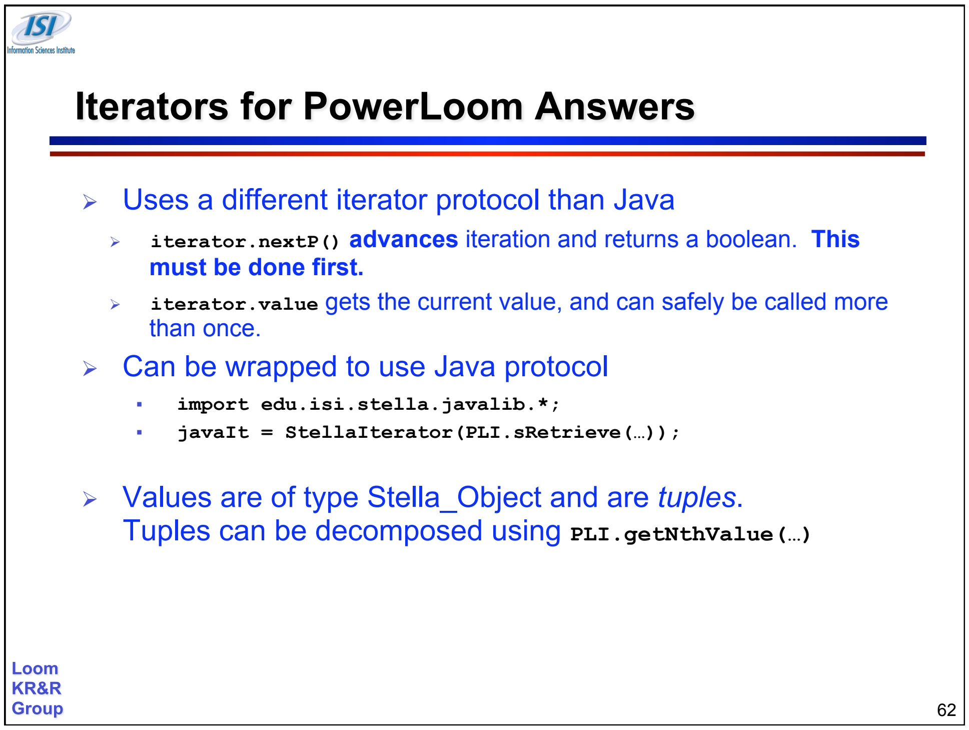
"Retrieve" Queries

- Retrieve queries return values of type **`PlIterator`**

```
String query = "all (and (Senator ?sen) (represents ?sen California)"
                + "(political-party ?sen ?party))";
PlIterator answer = PLI.sRetrieve(query, "POLITICS", null);

System.out.println("Answers to query `" + query + "'");
while (answer.nextP()) {  // Iterate over the answers
   System.out.println(answer.value);
}
```

Loom
KR&R
Group

61



Iterators for PowerLoom Answers

- Uses a different iterator protocol than Java
  - `iterator.nextP()` **advances** iteration and returns a boolean. **This must be done first.**
  - `iterator.value` gets the current value, and can safely be called more than once.
- Can be wrapped to use Java protocol
  - `import edu.isi.stella.javalib.*;`
  - `javaIt = StellaIterator(PLI.sRetrieve(…));`
- Values are of type Stella_Object and are *tuples*.
  Tuples can be decomposed using `PLI.getNthValue(…)`

Loom
KR&R
Group

62

PowerLoom Datatypes in Java

- Literals are returned wrapped but can be coerced.
  - `integer ⇒ int`
  - `float ⇒ double`
  - `string ⇒ String`
- Logic Objects
  - type is `edu.isi.powerloom.logic.LogicObject`
  - PowerLoom objects like relations, instances, descriptions, skolems
- Stella Objects
  - type is `edu.isi.stella.Stella_Object`
  - Most general type. Usually wrapped literals, but may be modules.

Loom
KR&R
Group

63

PowerLoom Datatypes Booby Traps

- Warning: You don't always get what you expect!
  - Skolems can appear when you expect, say, a number
  - Best to test the type first!

```
PLI.sAssertProposition("(and (age Fred 10) (> (weight Fred) 150))"…)

PlIterator answer;
answer = PLI.sRetrieve("1 (and (age Fred ?a) (weight Fred ?w))", …)

answer.nextP();

    // The next line works since age is 10, but is dangerous
int age = PLI.getNthInteger(answer.value, 0, "PL-USER", null);

    // The next line blows up because the answer is a skolem!
int weight = PLI.getNthInteger(answer.value, 1, "PL-USER", null);

if (PLI.isInteger(PLI.getNthValue(answer.value, 1, "PL-USER", null))) {
  weight = PLI.getNthInteger(answer.value, 1, "PL-USER", null);
}
```

Loom
KR&R
Group

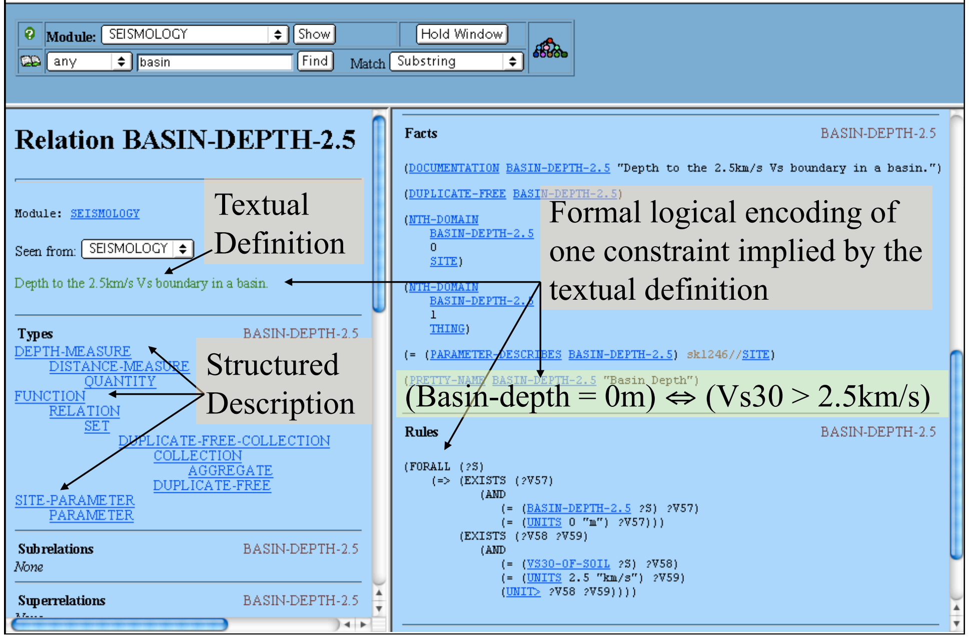
64

Additional Resources

- The interactive interface
  - Try things out before programming
- PowerLoom Manual
  - Has general information
  - Has information about Java-specific information
- Javadoc for PowerLoom
  - Caveat: For technical reasons almost all methods are public, but the intended API is contained mostly in the PLI class
- The example file PowerLoomExample.java
- PowerLoom website:
  http://www.isi.edu/isd/LOOM/PowerLoom/documentation

Ontosaurus:
Browsing PowerLoom

Ontosaurus Browser



Relation BASIN-DEPTH-2.5

Module: SEISMOLOGY

Seen from: SEISMOLOGY

Depth to the 2.5km/s Vs boundary in a basin.

Types BASIN-DEPTH-2.5

```
DEPTH-MEASURE
    DISTANCE-MEASURE
        QUANTITY
FUNCTION
    RELATION
        SET
            DUPLICATE-FREE-COLLECTION
                COLLECTION
                    AGGREGATE
                DUPLICATE-FREE
SITE-PARAMETER
    PARAMETER
```

Subrelations BASIN-DEPTH-2.5

None

Superrelations BASIN-DEPTH-2.5

Facts BASIN-DEPTH-2.5

```
(DOCUMENTATION BASIN-DEPTH-2.5 "Depth to the 2.5km/s Vs boundary in a basin.")

(DUPLICATE-FREE BASIN-DEPTH-2.5)

(NTH-DOMAIN
   BASIN-DEPTH-2.5
   0
   SITE)

(NTH-DOMAIN
   BASIN-DEPTH-2.5
   1
   THING)

(= (PARAMETER-DESCRIBES BASIN-DEPTH-2.5) sk1246//SITE)

(PRETTY-NAME BASIN-DEPTH-2.5 "Basin Depth")
```

Rules BASIN-DEPTH-2.5

```
(FORALL (?S)
   (=> (EXISTS (?V57)
          (AND
             (= (BASIN-DEPTH-2.5 ?S) ?V57)
             (= (UNITS 0 "m") ?V57)))
       (EXISTS (?V58 ?V59)
          (AND
             (= (VS30-OF-SOIL ?S) ?V58)
             (= (UNITS 2.5 "km/s") ?V59)
             (UNIT> ?V58 ?V59)))))
```

Textual Definition

Structured Description

Formal logical encoding of one constraint implied by the textual definition

(Basin-depth = 0m) $\Leftrightarrow$ (Vs30 > 2.5km/s)

Ontosaurus Demo

Loom
KR&R
Group

Conclusion

Loom
KR&R
Group

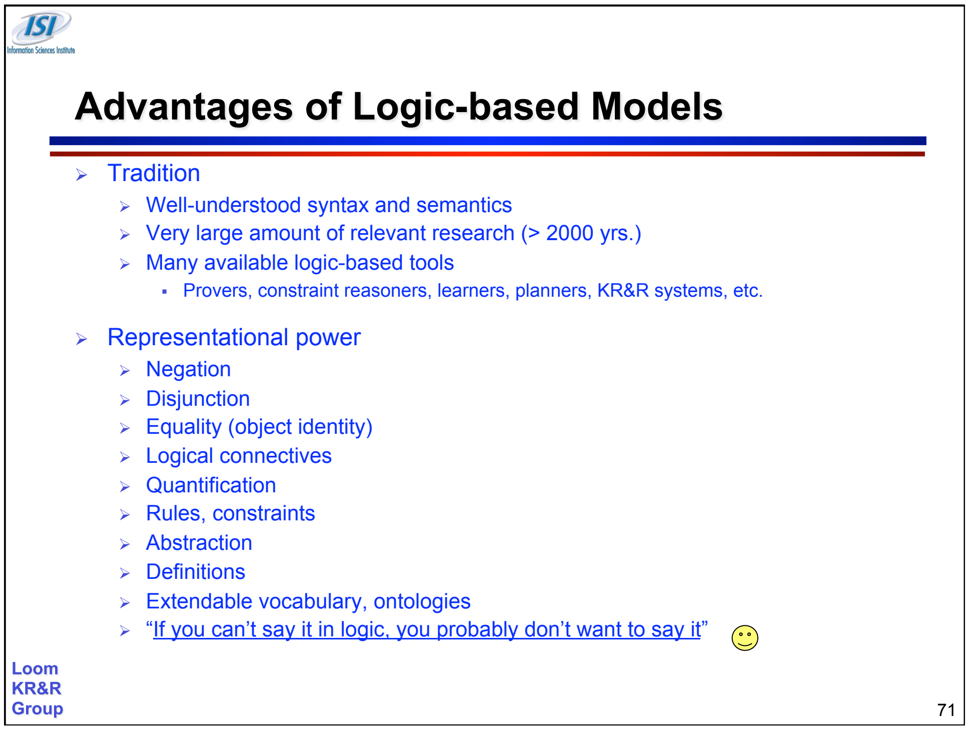
69

How Does Logic Model the World?

- Terms correspond to entities in the (some) world
- Predicates model properties and relations between entities
- Domain rules define and constrain relations, for example, "If Joe is a teenager who owns a car then Joe is happy"
- Logical inference rules define the propagation of truth between logical sentences, for example:
  from X and X => Y it must be true that Y
- The more rules and sentences we add, the higher constrained their "interpretation" (what they could mean) becomes
- However, every consistent theory always has infinitely many (formal) interpretations

Loom
KR&R
Group

70

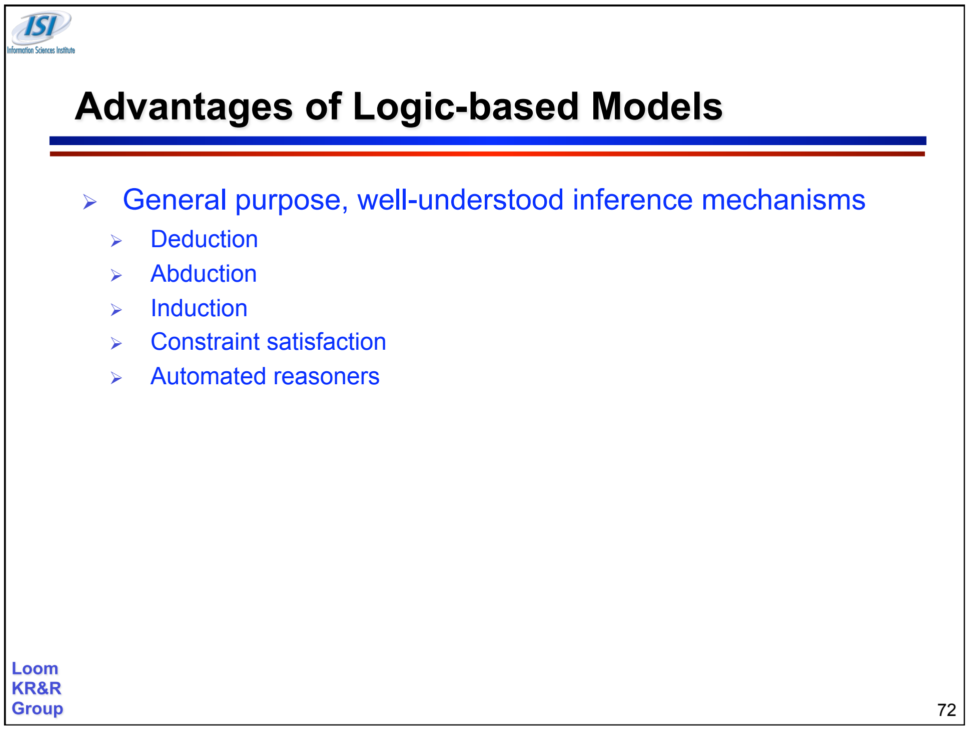


Advantages of Logic-based Models

- Tradition
  - Well-understood syntax and semantics
  - Very large amount of relevant research (> 2000 yrs.)
  - Many available logic-based tools
    - Provers, constraint reasoners, learners, planners, KR&R systems, etc.
- Representational power
  - Negation
  - Disjunction
  - Equality (object identity)
  - Logical connectives
  - Quantification
  - Rules, constraints
  - Abstraction
  - Definitions
  - Extendable vocabulary, ontologies
  - "If you can't say it in logic, you probably don't want to say it" ☺

Loom
KR&R
Group

71

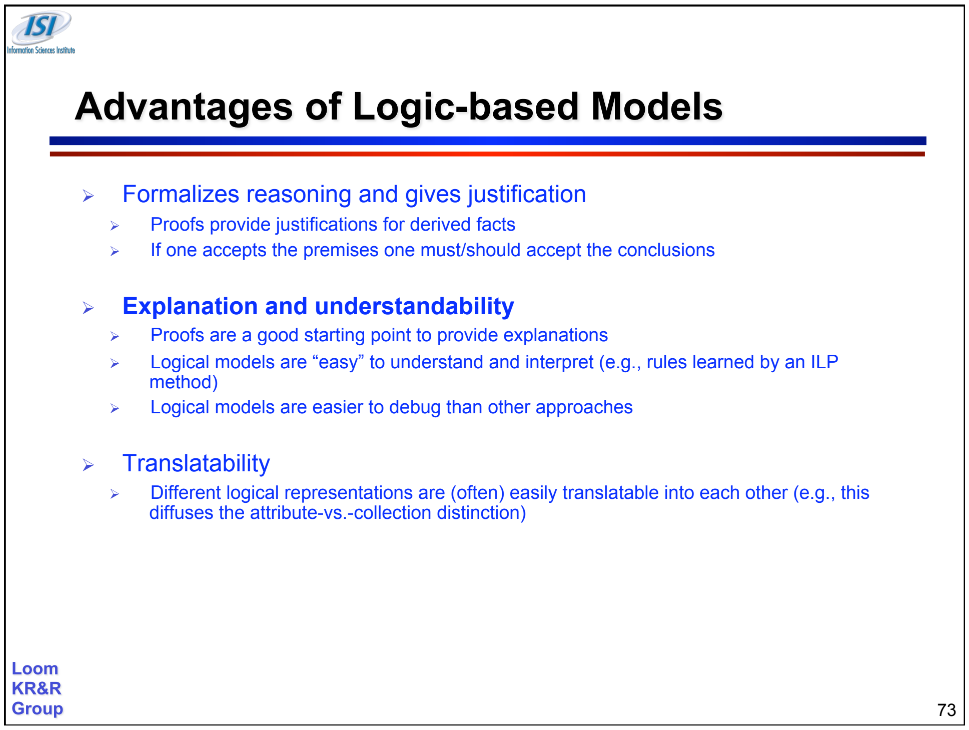


Advantages of Logic-based Models

- General purpose, well-understood inference mechanisms
  - Deduction
  - Abduction
  - Induction
  - Constraint satisfaction
  - Automated reasoners

Loom
KR&R
Group

72

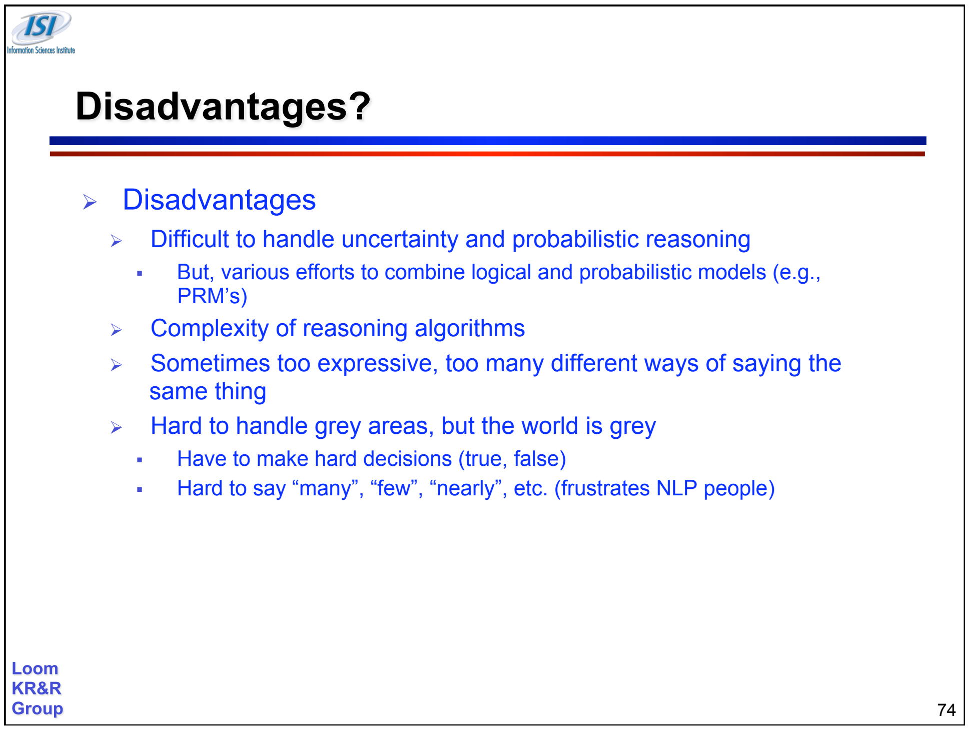


Advantages of Logic-based Models

- Formalizes reasoning and gives justification
  - Proofs provide justifications for derived facts
  - If one accepts the premises one must/should accept the conclusions
- **Explanation and understandability**
  - Proofs are a good starting point to provide explanations
  - Logical models are "easy" to understand and interpret (e.g., rules learned by an ILP method)
  - Logical models are easier to debug than other approaches
- Translatability
  - Different logical representations are (often) easily translatable into each other (e.g., this diffuses the attribute-vs.-collection distinction)

Loom KR&R Group

73



Disadvantages?

- Disadvantages
  - Difficult to handle uncertainty and probabilistic reasoning
    - But, various efforts to combine logical and probabilistic models (e.g., PRM's)
  - Complexity of reasoning algorithms
  - Sometimes too expressive, too many different ways of saying the same thing
  - Hard to handle grey areas, but the world is grey
    - Have to make hard decisions (true, false)
    - Hard to say "many", "few", "nearly", etc. (frustrates NLP people)

Loom KR&R Group

74