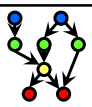


Lecture 17 of 42

Ontologies, Knowledge Engineering, and Elicitation of Domain Expertise Discussion: Actions, Situations, Events & Time

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

Reading for Next Class:
Section 10.3, p. 328 – 340, Russell & Norvig 2nd edition
Fluents: http://en.wikipedia.org/wiki/Fluent_%28artificial_intelligence%29
Situation calculus: http://en.wikipedia.org/wiki/Situation_calculus
KM: http://en.wikipedia.org/wiki/Knowledge_management

Lecture Outline

- **Reading for Next Class: Section 10.3 (p. 328 – 340), R&N 2e**
- **Last Week: Reasoning Architectures**
  - **Production systems: Rete algorithm (CLIPS, JESS, OPS5, *ArtEnterprise*)**
  - **Theorem provers: resolution (Prolog inference engine)**
  - **Ontology reasoners: Protégé-OWL, etc.**
- **Last Class: Prolog in Brief, Knowledge Engineering (KE), Ontologies**
  - **Prolog examples**
  - **Introduction to ontologies and ontology design**
    - **Description logics: $\mathcal{ALC}$**
    - **$\mathcal{SHOIN}$ (extended $\mathcal{ALC}$) and Web Ontology Language (OWL)**
- **Today: Knowledge Engineering (KE), Protocol Analysis, Fluents**
  - **Concept elicitation techniques: unstructured, structured, protocol analysis**
  - **Representing time, events: from situation calculus to event, fluent calculi**
  - **Preview: Knowledge acquisition (KA) and capture**
  - **Next: Computational information and knowledge management (CIKM)**
- **Coming Week: CIKM, Logical KR Concluded; Classical Planning**

Acknowledgements

Professor Ian Horrocks
Professor of Computer Science
Oxford University
Computing Laboratory
Fellow, Oriel College

© 2006
Horrocks, I.
Oxford University
(formerly University of Manchester)
http://bit.ly/ba4Jc

Natasha Noy
Senior Research Scientist
BMIR

Samson Tu
Senior Research Scientist
BMIR

© 2005
N. Noy & S. Tu
Stanford Center for Biomedical Informatics Research
http://bit.ly/jwOf3
http://bit.ly/2NBeCl
http://bmir.stanford.edu

Georghe Tecuci
Professor of Computer Science
Director, Learning Agents Center
George Mason University

© 2001
G. Tecuci
George Mason University
http://bit.ly/3tUACW
http://lalab.gmu.edu/cs785/
http://lac.gmu.edu

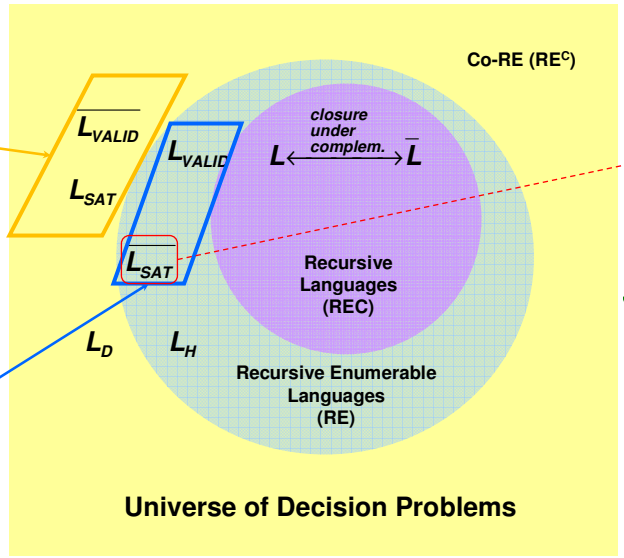
Decision Problems: Review

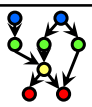
Undecidable duals
$\alpha \in L_{VALID}{}^{C}$ iff
$\neg\alpha \in L_{SAT}$

L_H: Halting problem

L_D: Diagonal problem

Semi-decidable duals:
$\alpha \in L_{VALID}$ iff
$\neg\alpha \in L_{SAT}{}^{C}$





DL Basics: Review

- Concepts (**formulae**)
  - E.g., Person, Doctor, HappyParent, (Doctor ⊔ Lawyer)
- Roles (**modalities**)
  - E.g., hasChild, loves
- Individuals (**nominals**)
  - E.g., John, Mary, Italy
- Operators (for forming concepts and roles) restricted so that:
  - Satisfiability/subsumption is decidable and, *if possible*, of low complexity
  - No need for explicit use of variables
    - Restricted form of ∃ and ∀ (**direct correspondence with ⟨i⟩ and [i]**)
  - Features such as counting (**graded modalities**) succinctly expressed

 http://bit.ly/10Oh4X

DL Knowledge Base: Example

- A TBox is a set of "schema" axioms (sentences), e.g.:

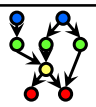
  {Doctor → Person,
  HappyParent ↔ Person ∧ [hasChild](Doctor ∨ ⟨hasChild⟩Doctor)}

  - i.e., a **background theory** (a set of **non-logical axioms**)
- An ABox is a set of "data" axioms (ground facts), e.g.:

  {John → HappyParent,
  John → ⟨hasChild⟩Mary}

  - i.e., non-logical axioms including (restricted) use of nominals
- A Knowledge Base (KB) is just a TBox plus an Abox

 http://bit.ly/10Oh4X



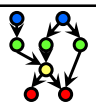
$\mathcal{ALC}$ Class / Concept Constructors: Review

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

- C is a concept (class); P is a role (property); x_i is an individual/nominal
- XMLS datatypes as well as classes in ∀P.C and ∃P.C
  - Restricted form of DL concrete domains

MANCHESTER 1824 img

 http://bit.ly/10Oh4X



$\mathcal{ALC}$ & $\mathcal{SHOIN}$/OWL Ontology Axioms: Review

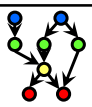
OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human ⊑ Animal ⊓ Biped
equivalentClass	$C_1 \equiv C_2$	Man ≡ Human ⊓ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter ⊑ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost ≡ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ ⊑ ancestor

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	⟨John, Mary⟩ : has-child

- OWL ontology equivalent to DL KB (Tbox + Abox)

MANCHESTER 1824 img

 http://bit.ly/10Oh4X



Description Logic Advantages: Review

- **OWL exploits results of 15+ years of DL research**
  - Well defined (model theoretic) **semantics**
  - **Formal properties** well understood (complexity, decidability)
  - Known **reasoning algorithms**
  - **Implemented systems** (highly optimised)

Cerebra® FaCT++ Racer MINDSWAP Pellet

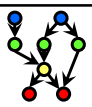
MANCHESTER 1824 iM.g

Adapted from slides © 2006 I. Horrocks, University of Manchester http://bit.ly/10Oh4X

What Is A "Concept"?

- "Concept" and "Class" are used synonymously
- A class is a concept in the domain
  - a class of wines
  - a class of wineries
  - a class of red wines
- A class is a collection of elements with similar properties
- Instances of classes
  - a glass of California wine you'll have for lunch

Adapted from slide © 2005 N. Noy & S. Tu
Stanford Center for Biomedical Informatics Research
http://bmir.stanford.edu



Class Inheritance

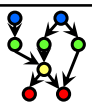
- Classes usually constitute a taxonomic hierarchy (a subclass-superclass hierarchy)
- A class hierarchy is usually an IS-A hierarchy:

an instance of a subclass is an instance of a superclass

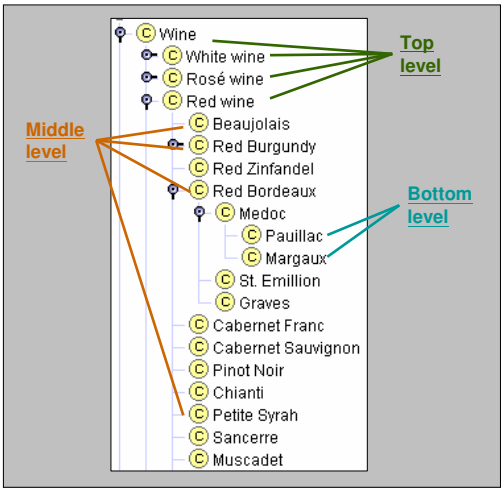
- If you think of a class as a set of elements, a subclass is a subset

Class Inheritance – Example

- Apple is a subclass of Fruit
  - *Every apple is a fruit*
- Red wine is a subclass of Wine
  - *Every red wine is a wine*
- Chianti wine is a subclass of Red wine
  - *Every Chianti wine is a red wine*



Levels In The Hierarchy



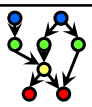
Defining Properties of Classes: Slots

- Slots, Attributes, and Relations: synonymous
- Slots in class definition *C*
  - *Describe attributes of instances of C*
  - *Describe relationships to other instances*
  - *e.g., each wine will have color, sugar content, producer, etc.*

Template Slots

Name	Type	Cardinality	Other Facets
body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
grape	Instance	multiple	classes={Wine grape}
maker	Instance	single	classes={Winery}
name	String	single	
sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

Slots for the Concept/Class *Wine*

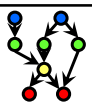


Concept Attributes: Properties & Slots

- Types of properties
  - "intrinsic" properties: flavor and color of wine
  - "extrinsic" properties: name and price of wine
  - parts: ingredients in a dish
  - relations to other objects: producer of wine (winery)
- Simple and complex properties
  - simple properties (attributes): contain primitive values (strings, numbers)
  - complex properties: contain (or point to) other objects (e.g., a winery instance)

Slot and Class Inheritance

- A subclass inherits all the slots from the superclass
  - *If a wine has a name and flavor, a red wine also has a name and flavor*
- If a class has multiple superclasses, it inherits slots from all of them
  - *Port is both a dessert wine and a red wine. It inherits "sugar content: high" from the former and "color:red" from the latter*



Property Constraints

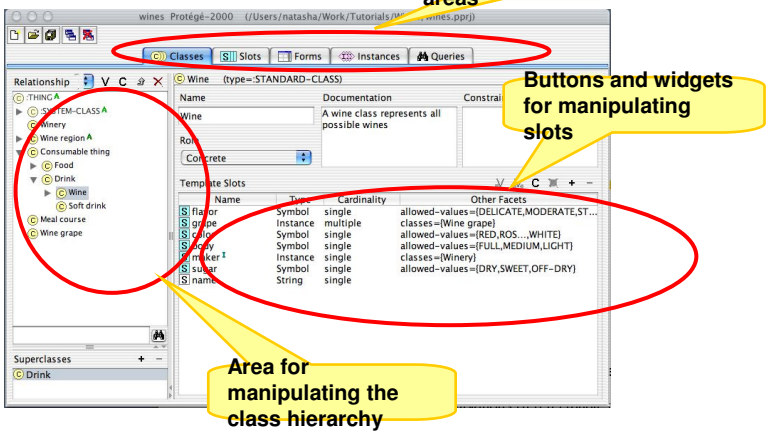
- Property constraints (facets) describe or limit the set of possible values for a slot
  - *The name of a wine is a string*
  - *The wine producer is an instance of Winery*
  - *A winery has exactly one location*

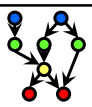
Template Slots

	Name	Type	Cardinality	Other Facets
S	body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
S	color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
S	flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
S	grape	Instance	multiple	classes={Wine grape}
S	maker	Instance	single	classes={Winery}
S	name	String	single	
S	sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

Facets for slots in the *Wine* class

Protégé: Default Interface





Protégé: Where to Go for Help

- Protégé user's guide
  - http://protege.stanford.edu/doc/users_guide/index.html
- Protégé 101 tutorial
  - http://bit.ly/178E6v
- FAQ
  - http://protege.stanford.edu/faq.html

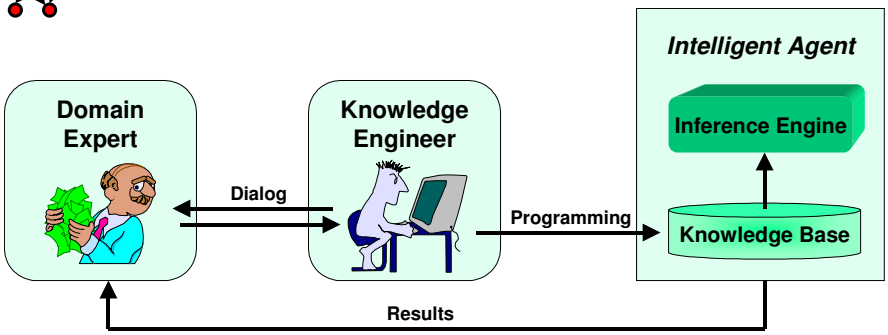
Adapted from slide © 2005 N. Noy & S. Tu
Stanford Center for Biomedical Informatics Research
http://bmir.stanford.edu

Knowledge Engineering: Overview

- A scenario for manual knowledge acquisition
- Elicitation of expert's conception of a domain
- Elicitation based on the personal construct theory
- Knowledge acquisition for role-limiting methods
- Advanced approaches to KB and agent development

CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/

How Agents Are Built



A knowledge engineer attempts to understand how a subject matter expert reasons and solves problems and then encodes the acquired expertise into the agent's knowledge base.

The expert analyzes the solutions generated by the agent (and often the knowledge base itself) to identify errors, and the knowledge engineer corrects the knowledge base.

CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/

Scenario for Manual Knowledge Acquisition

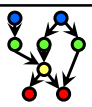
Adapted from:

B.G. Buchanan, D. Barstow, R. Bechtal, J. Bennett, W. Clancey, C. Kulikowski, T. Mitchell, D.A. Waterman,

Constructing an Expert System

in F. Hayes-Roth, D. Waterman and D. Lenat (*eds.*), *Building Expert Systems*, Addison-Wesley, 1983, pp. 127-168.

CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/



Problem Identification

The director of ORNL faces a problem. EPA regulations forbid the discharge of quantities of oil or hazardous chemicals into or upon waters of the United States, when this discharge violates specified quality standards. ORNL has approximately 2000 buildings on a 200 square mile government reservation, with 93 discharge sites entering White Oak Creek. Oil and hazardous chemicals are stored and used extensively at ORNL. The problem is to detect, monitor, and contain spills of these materials.

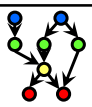
Investigated Solution

Develop a computer system that incorporates the expertise of people familiar with spill detection and containment (*i.e.,* a knowledge-based system, expert system or agent).

A knowledge engineer is assigned the job of building the system.

The knowledge engineer becomes familiar with the problem and the domain.

The knowledge engineer finds an expert on the subject who agrees to collaborate in building the system.



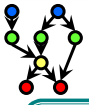
Requirements Specification: Scoping Problem to Solve

The knowledge engineer and the expert have a series of meetings to better identify the problem and to characterize it informally. They decide to concentrate on identifying, locating, and containing the spill:

When an accidental inland spill of an oil or chemical occurs, an emergency situation may exist, depending on the properties and quantity of the substance released, the location of the substance, and whether or not the substance enters a body of water.
The observer of a spill should:

1. **characterize the spill and the probable hazards,**
2. **contain the spill material,**
3. **locate the source of the spill and stop any further release,**
4. **notify the Department of Environmental Management.**

Based on slide © 2001 G. Tecuci, George Mason University
CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/



Understanding Domain of Expertise: Identifying Basic Domain Concepts [1]

The knowledge engineer schedules numerous meetings with the expert to uncover basic concepts, primitive relations, and definitions needed to talk about and understand this problem and its solutions. The following is a sample dialogue between the knowledge engineer and the expert:

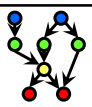
KE: Suppose you were told that a spill had been detected in White Oak Creek one mile before it enters White Oak Lake. What would you do to contain the spill?

SME: That depends on a number of factors. I would need to find the source in order to prevent the possibility of further contamination, probably by checking drains and manholes for signs of the spill material. And it helps to know what the spilled material is.

KE: How can you tell what it is?

SME: Sometimes you can tell what the substance is by its smell. Sometimes you can tell by its color, but that's not always reliable since dyes are used a lot nowadays. Oil, however, floats on the surface and forms a silvery film, while acids dissolves completely in the water. Once you discover the type of material spilled, you can eliminate any building that either don't store the material at all or don't store enough of it to account for the spill.

Adapted from slides © 2001 G. Tecuci, George Mason University
CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/



Understanding Domain of Expertise: Identifying Basic Domain Concepts [2]

As a result of such dialogues, the knowledge engineer identifies a set of concepts and features used in this problem:

- **Task:** Identification of spill material

- **Attributes of spill**
  - **Type of spill:** Oil, acid
  - **Location of spill:** <A set of drains and manholes>
  - **Volume of spill:** <A number of liters>

- **Attributes of material**
  - **Color:** Silvery, clear, etc.
  - **Odor:** Pungent/choking, etc.
  - **Does it dissolve?**
  - **Possible locations:** <A set of buildings>
  - **Amount stored:** <A number of liters>

Choosing System-Building Language or Tool

During conceptualization, the knowledge engineer also selects a general system-building language or tool for implementing the knowledge based system.

It was determined that the data are well-structured and fairly reliable and that the decision processes involve feedback and parallel decisions.

This suggests the use of a rule-based language.
Therefore the knowledge engineer decides to use the rule-based language ROSIE.

ROSIE provides a general (rule-based) inference engine, as well as a formalism for representing the knowledge in the form of assertions about objects and inference rules.

ROSIE could be regarded as a very general expert system shell.

Representing Domain Concepts: Object Ontology

The knowledge engineer attempts to represent the concepts in ROSIE's formalism:

```
ASSERT each of BUILDING 3023 and BUILDING 3024 is a building.
ASSERT s6-1 is a source in BUILDING 3023.
ASSERT s6-2 is a source in BUILDING 3024.
ASSERT s6-1 does hold 2000 gallons of gasoline.
ASSERT s6-2 does hold 50 gallons of acetic acid.
ASSERT each of d6-1 and d6-2 is a drain.
ASSERT each of m6-1 and m6-2 is a manhole.
ASSERT any drain is a location and any manhole is a location.
ASSERT each of diesel oil, hydraulic oil, transformer oil and gasoline is an oil.
ASSERT each of sulfuric acid, hydrochloric acid and acetic acid is an acid.
ASSERT every oil is a possible-material of the spill
        and every acid is a possible-material of the spill.
ASSERT the spill does smell of [some material, e.g. gasoline, vinegar, diesel oil].
ASSERT the spill does have [some odor, e.g., a pungent/choking, no] odor.
ASSERT the odor of the spill [is, is not] known.
ASSERT the spill does form [some appearance, e.g., a silvery film, no film].
ASSERT the spill [does, does not] dissolve in water.
```


Defining Problem-Solving Rules [1]

The knowledge engineer now uses the identified concepts to represent the expert's method of determining the spill material as a set of ROSIE rules:

To determine-spill-material:

[1] IF the spill does not dissolve in water
and the spill does form a silvery film,
THEN let the spill be oil.

[2] IF the spill does dissolve in water
and the spill does form no film,
THEN let the spill be acid.

(continued on next slide)

Defining Problem-Solving Rules [2]

```
[3]     IF      the spill = oil
                and the odor of the spill is known
        THEN    choose situation:

                IF      the spill does smell of gasoline
                THEN    let the material of the spill be gasoline with certainty .9;

                IF      the spill does smell of diesel oil
                THEN    let the material of the spill be diesel oil with certainty .8.

[4]     IF      the spill = acid
                and the odor of the spill is known,
        THEN    choose situation:

                IF      the spill does have a pungent/choking odor
                THEN    let the material of the spill be
                        hydrochloric acid with certainty .7;

                IF      the spill does smell of vinegar
                THEN    let the material of the spill be
                        acetic acid with certainty .8.

End.
```


Verifying Problem-Solving Rules

The knowledge engineer shows the rules to the expert and asks for reactions:

KE: Here are some rules I think capture your explanation about determining the type of material spilled and eliminating possible spill sources. What do you think?

SME: Uh-huh (long pause). Yes, that begins to capture it. Of course if the material is silver nitrate it will dissolve only partially in the water.

KE: I see. Well, let's add that information to the knowledge base and see what it looks like.

Refinement of Knowledge Base

The knowledge engineer may now revise the knowledge base by reformulating basic domain concepts, and refining the rules.

Delete: ASSERT the spill [does, does not] dissolve in water.

Add: ASSERT the solubility of the spill is
[some level - high, moderate, low].

Modify: [1]
IF the solubility of the spill is low
and the spill does form a silvery film,
THEN let the spill be oil.

Add: [1.5]
IF the solubility of the spill is moderate,
THEN let the material of the spill be silver-nitrate
with certainty .6

Adapted from slide © 2001 G. Tecuci, George Mason University
CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/

Main Phases of Agent Development Process

Defining problem to solve and system to be built: requirements specification

Understanding the expertise domain

Choosing or building an agent building tool: Inference engine and representation formalism

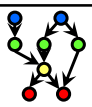
Development of the object ontology

Development of problem solving rules or methods

Refinement of the knowledge base

Feedback loops among all phases

Adapted from slide © 2001 G. Tecuci, George Mason University
CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/



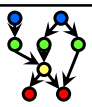
Elicitation of Expert's Conception of Domain

By eliciting the expert's conception of his/her expertise domain we mean determining which concepts apply in the domain, what they mean, what their relative place in the domain is, what the differentiating criteria are for distinguishing the similar concepts, and what the organizational structure is that keeps these concepts coherent for the expert.

Elicitation Methodology

(based primarily on Gammack, 1987)

1. **Concept elicitation: methods**
   (elicit concepts of domain, *i.e.* agreed-upon vocabulary)
2. **Structure elicitation: card-sort method**
   (elicit some structure for concepts)
3. **Structure representation**
   (formally represent structure in semantic network)
4. **Transformation of representation**
   (transform representation to be used for some desired purpose)



Concept Elicitation Methods [1]

Ask the expert to prepare an introductory talk outlining the whole domain, and to deliver it as a tutorial session to the knowledge engineer

Tape-record a lecture

Ask the expert to generate a list of typical concepts and then systematically probe for more relevant information (*e.g.*, using free association).

Identify concepts from the index of an expert's book

Adapted from slide © 2001 G. Tecuci, George Mason University
CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/

Concept Elicitation Methods [2]

Unstructured interview of the expert

Questions and alternative responses are open-ended

Example (the interview illustrated before in the "spill" application):

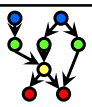
KE: Suppose you were told that a spill had been detected in White Oak Creek one mile before it enters White Oak Lake. What would you do to contain the spill?

SME: That depends on a number of factors. I would need to find the source in order to prevent the possibility of further contamination, probably by …

Used when the KE wants to explore an issue

Difficult to plan and conduct

Based on slide © 2001 G. Tecuci, George Mason University
CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/



Concept Elicitation Methods [3]

Structured interview of the expert

Questions are fixed in advance

Types of structured questions

- **Multiple-choice questions** *(offer specific choices, faster tabulation, and less bias due to the way the answers are ordered)*
- **Dichotomous** *(yes/no)* **questions**
- **Ranking scale questions** *(ask the expert to arrange items in a list in order of their importance or preference)*

Used when the KE wants specific information

Goal-oriented

Adapted from slide © 2001 G. Tecuci, George Mason University
CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/

Concept Elicitation Methods [4]

Protocol analysis (think-aloud technique)

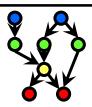
Systematic collection and analysis of the thought processes or problem-solving methods of an expert.

Protocols (cases, scenarios) are collected by asking experts to solve problems and to verbalize what goes through their minds, stating directly what they think. The solving process is carried out in an automatic fashion while the expert talks.

Knowledge engineer does not interrupt or ask questions.

Structuring the information elicited occurs later when the knowledge engineer analyzes the protocol.

Based on slide © 2001 G. Tecuci, George Mason University
CS 785 Knowledge Acquisition and Problem-Solving http://lalab.gmu.edu/cs785/



Concept Elicitation: Illustration [1]

Elicitation experiment in the domain of domestic gas-fired hot water and central heating system (Gammack, 1987).

Initial interview resulted in about 90 nouns or compound nouns, both concrete and abstract in nature.

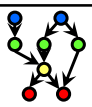
The expert edited this list by removing synonyms, slips of the tongue, and other aberrant terms, which reduced the list to 75 familiar concepts.

Concept Elicitation: Illustration [2]

The expert initially considered the dictionary definition of these concepts to be adequate, but since there is no guarantee that the expert's own definition necessarily matches the dictionary one, a personal definition of the concepts was given. This produced a few new concepts, such as "fluid", "safety", and "room".

The definitions indicated that sometimes a concept went beyond the level of detail given in a general purpose dictionary and sometimes it meant one very specific idea in the context of the domain.

This illustrates an important issue:
Much human expertise is likely to consist in the personal and semantic associations (connotative meaning) that an expert brings to domain concepts and may result in the invention or appropriation of personalized terms to describe esoteric or subtle domain phenomena.



Concept Elicitation: Illustration [3]

The domain glossary obtained characterized the component parts of a central heating system, such as thermostats and radiators, but also included general physical terms such as heat and gravity.

A second path through the transcript yielded 42 relational concepts, usually verbs (contains, heats, connects to, *etc.*). These concepts will be used later to label relationships between the discovered concepts.

Concept Elicitation Methods: Features

Strengths

- gives the knowledge engineer an orientation to the domain.
- generates much knowledge cheaply and naturally.
- not a significant effort for the expert.

Weaknesses

- incomplete and arbitrary coverage
- the knowledge engineer needs appropriate training and/or social skills

Handling Actions, Situations, Time: Situation, Event & Fluent Calculi

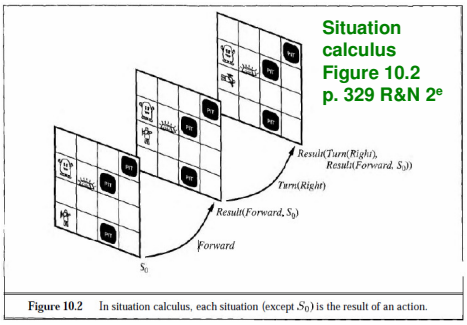
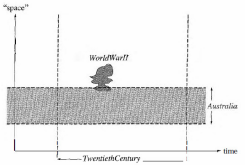


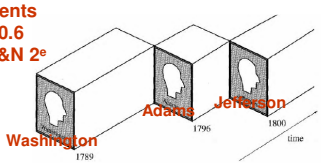
Figure 10.2 In situation calculus, each situation (except S_0) is the result of an action.

Situation calculus
Figure 10.2
p. 329 R&N 2e



Event calculus
Figure 10.3
p. 336 R&N 2e

State fluents
Figure 10.6
p. 340 R&N 2e

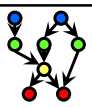


Figures © 2003 S. Russell & P. Norvig. Reused with permission.

- Event Calculus: Reasoning about Entities over Time, Space
- Fluent Calculus: Variant of Situation Calculus
  - Conditions (predicates) that can change over time
  - Defaults
  - ∘ (concatenation)

Preview: Computational Information & Knowledge Management

- Information Management
  - Data acquisition: instrumentation, collection, polling, elicitation
  - Data and information integration: combining multiple sources
    - ⇨ May be heterogeneous (different in quality, format, rate, *etc.*)
    - ⇨ Underlying formats, properties may correspond to different ontologies
    - ⇨ Ontology mappings (functions to convert between ontologies) needed
  - Data transformation: preparation for reasoning, learning
    - ⇨ Preprocessing
    - ⇨ Cleaning
  - Includes knowledge capture: assimilation from various sources
- Knowledge Management
  - Term used most often in business administration, management science
  - Related to IM, but capability and process-centered
  - Focus on learning and KA, organization theory, decision theory
    - ⇨ Discussion, apprenticeship, forums, libraries, training/mentoring
    - ⇨ Modern theory: KBs, Expert Systems, Decision Support Systems



Terminology

- Knowledge Engineering (KE): Process of KR Design, Acquisition
  - Knowledge
    - What agents possess (epistemology) that lets them reason
    - Basis for rational cognition, action
    - Knowledge gain (acquisition, learning): improvement in problem solving
  - Knowledge level (*vs.* symbol level): level at which agents reason
  - Semantic network: inheritance and membership/containment relationships
  - Knowledge elicitation: KA/KE process from human domain experts
    - Protocol analysis: preparing, conducting, interpreting interview
    - Less formal methods: subjective estimation & probabilities
- Fluents: Conditions (Predicates) That Can Change over Time
  - Classes, nominals (objects / class instances): spatial, temporal extent
  - Fluent calculus: situation calculus with defaults, ∘ (concatenation)
- Computational Information and Knowledge Management (CIKM)
  - Data/info integration & transformation: collecting, preparing data
  - Includes knowledge capture: assimilation from various sources

Summary Points

- Last Class: Prolog in Brief, Description Logics, Ontologies
  - Prolog examples
  - Ontologies: formal languages for describing domains for KR
  - KR as basis of learning and reasoning
  - *ALC*, *SHOIN*, and Web Ontology Language (OWL)
- Today: More Ontology Design; Knowledge Engineering, Elicitation
  - Concept elicitation techniques
    - Unstructured
    - Structured
    - Protocol analysis
  - Knowledge acquisition (KA); info and knowledge management defined
  - Situation calculus revisited; time and event calculus, fluent calculus
- Next Class: More KE, Semantic Nets
  - KA and knowledge capture: elicitation concluded (structure elicitation)
  - Computational information and knowledge management (CIKM)
- Coming Week: Logical KR Concluded; Planning