CSC 480: Artificial Intelligence

Dr. Franz J. Kurfess
Computer Science Department
Cal Poly

Course Overview

- Introduction
- Intelligent Agents
- Search
  - problem solving through search
  - informed search
- Games
  - games as search problems
- **Knowledge and Reasoning**
  - reasoning agents
  - **propositional logic**
  - **predicate logic**
  - knowledge-based systems
- Learning
  - learning from observation
  - neural networks
- Conclusions

Logistics - Nov. 8, 2012

- **AI Nugget presentations scheduled**
  - Section 1:
    - none
  - Section 3:
    - Bryan Stoll: Virtual Composer (delayed from Oct. 25)
    - Spencer Lines: What IBM's Watson has been up to since it won in 2011
    - Marcus Jackson: Creating an Artificial Human Brain
    - Luke Diedrich: Artificial intelligence with Quadrocopters
    - Jennifer Gaona: Neural Networks in Prosthetics (postponed to Nov. 8)
- **Quiz**
  - Quiz 7 - Reasoning & Logic
- **Labs**
  - Lab 8 due Tue, Nov 13: Reasoning and Knowledge in the Wumpus World (Web form)
    - related to A2 Part 1
- **A2 Wumpus World**
  - Part 1: Knowledge Representation and Reasoning
    - Web form, no programming required
    - Due: today
  - Part 2: Implementation
    - Due: Nov. 15
- **A3 Competitions converted to optional**
  - weight of remaining assignments adjusted accordingly

Chapter Overview
Logic

- Motivation
- Objectives
- Propositional Logic
  - syntax
  - semantics
  - validity and inference
  - models
  - inference rules
  - complexity
  - limitations
  - Wumpus agents
- Predicate Logic
  - Principles
    - objects
    - relations
    - properties
- Syntax
- Semantics
- Extensions and Variations
- Usage
  - Logic and the Wumpus World
    - reflex agent
    - change
- Important Concepts and Terms
- Chapter Summary

Motivation

- formal methods to perform reasoning are required when dealing with knowledge
- propositional logic is a simple mechanism for basic reasoning tasks
  - it allows the description of the world via sentences
    - simple sentences can be combined into more complex ones
    - new sentences can be generated by inference rules applied to existing sentences
- predicate logic is more powerful, but also considerably more complex
  - it is very general, and can be used to model or emulate many other methods
  - although of high computational complexity, there is a subclass that can be treated by computers reasonably well

Objectives

- know the important aspects of propositional and predicate logic
  - syntax, semantics, models, inference rules, complexity
- understand the limitations of propositional and predicate logic
- apply simple reasoning techniques to specific tasks
- learn about the basic principles of predicate logic
- apply predicate logic to the specification of knowledge-based systems and agents
- use inference rules to deduce new knowledge from existing knowledge bases

Logical Inference

- also referred to as **deduction**
  - implements the **entailment** relation for sentences
    - operates at the *semantic* level
    - takes into account the *meaning* of sentences
  - computers have difficulties reasoning at the semantic level
    - typically work at the *syntactic* level
    - *derivation* is used to approximate entailment
    - uses purely “mechanical” symbol manipulation without consideration of meaning
    - should be used with care since more constraints apply

Validity and Satisfiability

- validity
  - a sentence is **valid** if it is true under all possible interpretations in all possible world states
    - independent of its intended or assigned meaning
    - independent of the state of affairs in the world under consideration
    - valid sentences are also called tautologies
- satisfiability
  - a sentence is **satisfiable** if there is some interpretation in some world state (a *model*) such that the sentence is true
- relationship between satisfiability and validity
  - a sentence is satisfiable iff (“if and only if”) its negation is not valid
  - a sentence is valid iff its negation is not satisfiable

Computational Inference

- computers cannot reason informally ("common sense")
  - they don't know the interpretation of the sentences
  - they usually don't have access to the state of the real world to check the correspondence between sentences and facts
- computers can be used to check the validity of sentences
  - "if the sentences in a knowledge base are true, then the sentence under consideration must be true, regardless of its possible interpretations"
  - can be applied to rather complex sentences

Computational Approaches to Inference

- model checking based on truth tables
  - generate all possible models and check them for validity or satisfiability
  - exponential complexity, NP-complete
    - all combinations of truth values need to be considered
- search
  - use inference rules as successor functions for a search algorithm
  - also exponential, but only worst-case
    - in practice, many problems have shorter proofs
    - only relevant propositions need to be considered

Propositional Logic

- a relatively simple framework for reasoning
- can be extended for more expressiveness at the cost of computational overhead
- important aspects
  - syntax
  - semantics
  - validity and inference
  - models
  - inference rules
  - complexity

Truth Tables for Connectives

P	Q	$\neg P$	$P \land Q$	$P \lor Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Validity and Inference

- truth tables can be used to test sentences for validity
  - one row for each possible combination of truth values for the symbols in the sentence
  - the final value must be `True` for every sentence
  - a variation of the model checking approach
  - in general, not very practical for large sentences
    - can be very effective with customized improvements in specific domains, such as VLSI design

Validity Example

- known facts about the Wumpus World
  - there is a wumpus in [1,3] or in [2,2]
  - there is no wumpus in [2,2]
- question (hypothesis)
  - is there a wumpus in [1,3]
- task
  - prove or disprove the validity of the question
- approach
  - construct a sentence that combines the above statements in an appropriate manner
    - so that it answers the questions
  - construct a truth table that shows if the sentence is valid
    - incremental approach with truth tables for sub-sentences

Validity Example

P	Q	$P \lor Q$
False	False	False
False	True	True
True	False	True
True	True	True

W_{13}	$\lor$	W_{22}	$W_{13} \lor W_{22}$
False		False True	False
False		False	True
True		True	True
True			True

Interpretation:

W_{13} Wumpus in [1,3]

W_{22} Wumpus in [2,2]

Facts:

- there is a wumpus in [1,3] or in [2,2]

Validity Example

P	Q	$P \wedge Q$
False	*False*	*False*
False	*True*	*False*
True	*False*	*False*
True	*True*	*True*

$W_{13} \vee W_{22}$		$\neg W_{22}$
False	$\wedge$	*True*
True		*False*
True		*True*
True		*False*

Interpretation:

W_{13} Wumpus in [1,3]

W_{22} Wumpus in [2,2]

Facts:

- there is a wumpus in [1,3] or in [2,2]
- there is no wumpus in [2,2]

Validity Example

$W_{13} \vee W_{22}$
False
True
True
True

$\wedge$

$\neg W_{22}$
True
False
True
False

P	Q	$P \Rightarrow Q$
False	False	True
False	True	True
True	False	False
True	True	True

$(W_{13} \vee W_{22}) \wedge \neg W_{22}$
False
False
True
False

$\Rightarrow$

W_{13}
False
False
True
True

Question:

- can we conclude that the wumpus is in [1,3]?

Validity Example

$W_{13} \lor W_{22}$
False
True
True
True

$\land$

$\neg W_{22}$
True
False
True
False

$(W_{13} \lor W_{22}) \land \neg W_{22}$
False
False
True
False

$\Rightarrow$

W_{13}
False
False
True
True

$((W_{13} \lor W_{22}) \land \neg W_{22}) \Rightarrow W_{13}$
True
True
True
True

Valid Sentence:
For all possible combinations, the value of the sentence is `true.`

Validity and Computers

- the computer may not have access to the real world, to check the truth value of sentences (facts)
  - humans often can do that, which greatly decreases the complexity of reasoning
  - humans also have experience in considering only important aspects, neglecting others
- if a conclusion can be drawn from premises, independent of their truth values, then the sentence is valid
  - usually too tedious for humans
  - may exclude potentially interesting sentences
    - where some, but not all interpretations are true

Models

- if there is an interpretation for a sentence such that the sentence is true in a particular world, that world is called a **model**
  - refers to specific interpretations
- models can also be thought of as mathematical objects
  - these mathematical models can be viewed as equivalence classes for worlds that have the truth values indicated by the mapping under that interpretation
  - a model then is a mapping from proposition symbols to `True` or `False`

Models and Entailment

- a sentence α is **entailed** by a knowledge base KB if the models of the knowledge base KB are also models of the sentence α

$$KB \models \alpha$$

  - reasoning at the semantic level

Inference and Derivation

- inference rules allow the construction of new sentences from existing sentences
  - notation: a sentence β can be derived from α

$$\alpha \vdash \beta \quad \text{or} \quad \frac{\alpha}{\beta}$$

- an **inference procedure** generates new sentences on the basis of inference rules
- if all the new sentences are entailed, the inference procedure is called **sound** or **truth-preserving**

Inference Rules

- **modus ponens**
  - from an implication and its premise one can infer the conclusion

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

- **and-elimination**
  - from a conjunct, one can infer any of the conjuncts

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

- **and-introduction**
  - from a list of sentences, one can infer their conjunction

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

- **or-introduction**
  - from a sentence, one can infer its disjunction with anything else

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

Inference Rules

- **double-negation elimination**
  - a double negations infers the positive sentence
- **unit resolution**
  - if one of the disjuncts in a disjunction is false, then the other one must be true
- **resolution**
  - β cannot be true and false, so one of the other disjuncts must be true
  - can also be restated as "implication is transitive"

$$\frac{\neg \neg \alpha}{\alpha}$$

$$\frac{\alpha \lor \beta, \quad \neg \beta}{\alpha}$$

$$\frac{\alpha \lor \beta, \quad \neg \beta \lor \gamma}{\alpha \lor \gamma}$$

$$\frac{\neg \alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg \alpha \Rightarrow \gamma}$$

Tuesday, November 6, 12

Complexity

- the truth-table method to inference is complete
  - enumerate the 2^n rows of a table involving *n* symbols
  - computation time is exponential
- satisfiability for a set of sentences is NP-complete
  - so most likely there is no polynomial-time algorithm
  - in many practical cases, proofs can be found with moderate effort
- there is a class of sentences with polynomial inference procedures (Horn sentences or Horn clauses)

$$P_1 \land P_2 \land \ldots \land P_n \Rightarrow Q$$

Wumpus Logic

- an agent can use propositional logic to reason about the Wumpus world
  - knowledge base contains
    - percepts
    - rules

$\neg S_{1,1}$
$\neg S_{2,1}$
$S_{1,2}$
$\neg B_{1,1}$
$B_{2,1}$
$\neg B_{1,2}$

R1: $\neg S_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

R2: $\neg S_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$

R3: $\neg S_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$

R4: $S_{1,2} \Rightarrow W_{1,1} \vee W_{1,2} \vee W_{2,2} \vee W_{1,3}$

. . .

Finding the Wumpus

- two options
  - construct truth table to show that $W_{1,3}$ is a valid sentence
    - rather tedious
  - use inference rules
    - apply some inference rules to sentences already in the knowledge base

Action in the Wumpus World

- additional rules are required to determine actions for the agent

$$\mathbf{RM:}\ \mathbf{A}_{1,1} \wedge \mathbf{East}_A \wedge W_{2,1} \Rightarrow \neg\ Forward_A$$
$$\mathbf{RM+1:}\quad \ldots$$
$$\ldots$$

- the agent also needs to **ASK** the knowledge base what to do
  - must ask specific questions
    - Can I go forward?
  - general questions are not possible in propositional logic
    - Where should I go?

Propositional Wumpus Agent

- the size of the knowledge base even for a small wumpus world becomes immense
  - explicit statements about the state of each square
  - additional statements for actions, time
  - easily reaches thousands of sentences
- completely unmanageable for humans
- efficient methods exist for computers
  - optimized variants of search algorithms
  - sequential circuits
    - combinations of gates and registers
    - more efficient treatment of time
    - effectively a reflex agent with state
    - can be implemented in hardware

Exercise: Wumpus World in Propositional Logic

- express important knowledge about the Wumpus world through sentences in propositional logic format
  - status of the environment
  - percepts of the agent in a specific situation
  - new insights obtained by reasoning
    - rules for the derivation of new sentences
    - new sentences
  - decisions made by the agent
  - actions performed by the agent
    - changes in the environment as a consequence of the actions
  - background
    - general properties of the Wumpus world
  - learning from experience
    - general properties of the Wumpus world

Limitations of Propositional Logic

- number of propositions
  - since everything has to be spelled out explicitly, the number of rules is immense
- dealing with change (monotonicity)
  - even in very simple worlds, there is change
  - the agent's position changes
  - time-dependent propositions and rules can be used
    - even more propositions and rules
- propositional logic has only one representational device, the proposition
  - difficult to represent objects and relations, properties, functions, variables, ...

Bridge-In to Predicate Logic

- limitations of propositional logic in the Wumpus World
  - enumeration of statements
  - change
  - proposition as representational device
- usefulness of objects and relations between them
  - properties
  - internal structure
  - arbitrary relations
  - functions

Knowledge Representation and Commitments

- ontological commitment
  - describes the basic entities that are used to describe the world
    - e.g. facts in propositional logic
- epistemological commitment
  - describes how an agent expresses its believes about facts
    - e.g. true, false, unknown in propositional logic

Formal Languages and Commitments

Language	*Ontological Commitment*	*Epistemological Commitment*
Propositional Logic	facts	true, false, unknown
First-order Logic	facts, objects, relations	true, false, unknown
Temporal Logic	facts, objects, relations, times	true, false, unknown
Probability Theory	facts	degree of belief $\in [0,1]$
Fuzzy Logic	facts with degree of truth $\in [0,1]$	known interval value

Commitments in FOL

- ontological commitments
  - facts
    - same as in propositional logic
  - objects
    - corresponds to entities in the real world (physical objects, concepts)
  - relations
    - connects objects to each other
- epistemological commitments
  - true, false, unknown
    - same as in propositional logic

Predicate Logic

- new concepts
  - complex objects
    - terms
  - relations
    - predicates
    - quantifiers
  - syntax
  - semantics
  - inference rules
  - usage

Examples of Objects, Relations

- “The smelly wumpus occupies square [1,3]”
  - objects: wumpus, $square_{1,3}$
  - property: smelly
  - relation: occupies
- “Two plus two equals four”
  - objects: two, four
  - relation: equals
  - function: plus

Objects

- distinguishable things in the real world
  - e.g. people, cars, computers, programs, ...
  - the set of objects determines the domain of a model
- frequently includes concepts
  - colors, stories, light, money, love, ...
  - in contrast to *physical* objects
- properties
  - describe specific aspects of objects
    - green, round, heavy, visible,
  - can be used to distinguish between objects

Relations

- establish connections between objects
  - unary relations refer to a single object
    - e.g. `mother-of(John), brother-of(Jill), spouse-of(Joe)`
    - often called functions
  - binary relations relate two objects to each other
    - e.g. `twins(John,Jill), married(Joe, Jane)`
  - *n*-ary relations relate *n* objects to each other
    - e.g. `triplets(Jim, Tim, Wim), seven-dwarfs(D1, ..., D7)`
- relations can be defined by the designer or user
  - neighbor, successor, next to, taller than, younger than, …
- functions are a special type of relation
  - non-ambiguous: only one output for a given input
  - often distinguished from similar binary relations by appending `-of`
    - e.g. `brothers(John, Jim)` vs. `brother-of(John)`

Syntax

- based on sentences
  - more complex than propositional logic
    - constants, predicates, terms, quantifiers
- constant symbols

  `A, B, C, Franz,` $Square_{1,3}$`, ...`
  - stand for unique objects (in a specific context)
- predicate symbols

  `Adjacent-To, Younger-Than, ...`
  - describes relations between objects
- function symbols

  `Father-Of, Square-Position, …`
  - the given object is related to exactly one other object

Semantics

- relates sentences to *models*
  - in order to determine their truth values
- provided by *interpretations* for the basic constructs
  - usually suggested by meaningful names (intended interpretations)
- constants
  - the interpretation identifies the object in the real world
- predicate symbols
  - the interpretation specifies the particular relation in a model
  - may be explicitly defined through the set of tuples of objects that satisfy the relation
- function symbols
  - identifies the object referred to by a tuple of objects
  - may be defined implicitly through other functions, or explicitly through tables

BNF Grammar Predicate Logic

Sentence	→ *AtomicSentence*		
	| *(Sentence Connective Sentence)*		
	| *Quantifier Variable, ... Sentence*		
	| ¬ *Sentence*		
AtomicSentence	→ *Predicate(Term, ...)*	| *Term = Term*	
Term	→ *Function(Term, ...)*	| *Constant*	| *Variable*
Connective	→ ∧ | ∨ | ⇒ | ⇔		
Quantifier	→ ∀ | ∃		
Constant	→ *A, B, C,* X_1, X_2, *Jim, Jack*		
Variable	→ *a, b, c,* x_1, x_2, *counter, position*		
Predicate	→ *Adjacent-To, Younger-Than,*		
Function	→ *Father-Of, Square-Position, Sqrt, Cosine*		

Terms

- logical expressions that specify objects
- constants and variables are terms
- more complex terms are constructed from function symbols and simpler terms, enclosed in parentheses
  - basically a complicated name of an object
- semantics is constructed from the basic components, and the definition of the functions involved
  - either through explicit descriptions (e.g. table), or via other functions

Atomic Sentences

- state facts about objects and their relations
- specified through predicates and terms
  - the predicate identifies the relation, the terms identify the objects that have the relation
- an atomic sentence is `true` if the relation between the objects holds
  - this can be verified by looking it up in the set of tuples that define the relation

Examples Atomic Sentences

- `Father(Jack, John)`
- `Mother(Jill, John)`
- `Sister(Jane, John)`
- `Parents(Jack, Jill, John, Jane)`
- `Married(Jack, Jill)`
- `Married(Father-Of(John), Mother-Of(John))`
- `Married(Father-Of(John), Mother-Of(Jane))`
- `Married(Parents(Jack, Jill, John, Jane))`

Complex Sentences

- logical connectives can be used to build more complex sentences
- semantics is specified as in propositional logic

Examples Complex Sentences

- Father(Jack, John) ∧ Mother(Jill, John) ∧ Sister(Jane, John)
- ¬ Sister(John, Jane)
- Parents(Jack, Jill, John, Jane) ∧ Married(Jack, Jill)
- Parents(Jack, Jill, John, Jane) ⇒ Married(Jack, Jill)
- Older-Than(Jane, John) ∨ Older-Than(John, Jane)
- Older(Father-Of(John), 30) ∨ Older (Mother-Of(John), 20)

Attention: Some sentences may look like tautologies, but only because we "automatically" assume the meaning of the name as the only interpretation (parasitic interpretation)

Quantifiers

- can be used to express properties of collections of objects
  - eliminates the need to explicitly enumerate all objects
- predicate logic uses two quantifiers
  - universal quantifier $\forall$
  - existential quantifier $\exists$

Universal Quantification

- states that a predicate *P* is holds for all objects *x* in the universe under discourse
  $\forall x\ P(x)$
- the sentence is `true` if and only if all the individual sentences where the variable *x* is replaced by the individual objects it can stand for are `true`

Example Universal Quantification

- assume that *x* denotes the squares in the wumpus world
  - $\forall x$ *Is-Empty(x)* $\lor$ *Contains-Agent(x)* $\lor$ *Contains-Wumpus(x)* is true if and only if all of the following sentences are true:

    Is-empty$(S_{11}) \lor$ *Contains-Agent*$(S_{11}) \lor$ *Contains-Wumpus*(S_{11})

    Is-empty$(S_{12}) \lor$ *Contains-Agent*$(S_{12}) \lor$ *Contains-Wumpus*(S_{12})

    Is-empty$(S_{13}) \lor$ *Contains-Agent*$(S_{13}) \lor$ *Contains-Wumpus*(S_{13})

    . . .

    Is-empty$(S_{21}) \lor$ *Contains-Agent*$(S_{21}) \lor$ *Contains-Wumpus*(S_{21})

    . . .

    Is-empty$(S_{44}) \lor$ *Contains-Agent*$(S_{44}) \lor$ *Contains-Wumpus*(S_{44})

  - beware the implicit (parasitic) interpretation fallacy!

Usage of Universal Qualification

- universal quantification is frequently used to make statements like “All humans are mortal”, “All cats are mammals”, “All birds can fly”, …
- this can be expressed through sentences like

  $\forall x \; Human(x) \Rightarrow Mortal(x)$

  $\forall x \; Cat(x) \Rightarrow Mammal(x)$

  $\forall x \; Bird(x) \Rightarrow Can\text{-}Fly(x)$
- these sentences are equivalent to the explicit sentence about individuals

  $Human(John) \Rightarrow Mortal(John) \wedge$

  $Human(Jane) \Rightarrow Mortal(Jane) \wedge$

  $Human(Jill) \Rightarrow Mortal(Jill) \wedge \quad . \; . \; .$

Existential Quantification

- states that a predicate P holds for some objects in the universe
  $\exists x \; P(x)$
- the sentence is `true` if and only if there is at least one `true` individual sentence where the variable *x* is replaced by the individual objects it can stand for

Example Existential Quantification

- assume that *x* denotes the squares in the wumpus world
  - $\exists\, x\ Glitter(x)$ is true if and only if at least one of the following sentences is true:

    $Glitter(S_{11})$

    $Glitter(S_{12})$

    $Glitter(S_{13})$

    . . .

    $Glitter(S_{21})$

    . . .

    $Glitter(S_{44})$

Usage of Existential Qualification

- existential quantification is used to make statements like
  "Some humans are computer scientists",
  "John has a sister who is a computer scientist"
  "Some birds can't fly", …
- this can be expressed through sentences like

  $\exists\, x \;\; Human(x) \wedge Computer\text{-}Scientist(x)$

  $\exists\, x \;\; Sister(x, John) \wedge Computer\text{-}Scientist(x)$

  $\exists\, x \;\; Bird(x) \wedge \neg\, Can\text{-}Fly(x)$
- these sentences are equivalent to the explicit sentence about individuals

  $Human(John) \wedge \neg\, Computer\text{-}Scientist(John) \;\vee$

  $Human(Jane) \wedge Computer\text{-}Scientist(Jane) \;\vee$

  $Human(Jill) \wedge \neg\, Computer\text{-}Scientist(Jill) \;\vee$

Multiple Quantifiers

- more complex sentences can be formulated by multiple variables and by nesting quantifiers
  - the order of quantification is important
  - variables must be introduced by quantifiers, and belong to the innermost quantifier that mention them
  - examples

    $\forall x, y\ \ Parent(x,y) \Rightarrow Child(y,x)$

    $\forall x\ Human(x)\ \exists\ y\ \ Mother(y,x)$

    $\forall x\ Human(x)\ \exists\ y\ \ Loves(x, y)$

    $\exists\ x\ Human(x)\ \forall\ y\ \ Loves(x, y)$

    $\exists\ x\ Human(x)\ \forall\ y\ \ Loves(y,x)$

Connections between ∀ and ∃

- all statements made with one quantifier can be converted into equivalent statements with the other quantifier by using negation
  - ∀ is a conjunction over all objects under discourse
  - ∃ is a disjunction over all objects under discourse
  - De Morgan's rules apply to quantified sentences

$\forall x \neg P(x) \equiv \neg \exists x \ P(x)$ $\qquad$ $\neg \forall x \ P(x) \equiv \exists x \ \neg P(x)$

$\forall x \ P(x) \equiv \neg \exists x \ \neg P(x)$ $\qquad$ $\neg \forall x \ \neg P(x) \equiv \exists x \ P(x)$

- strictly speaking, only one quantifier is necessary
  - using both is more convenient

Domains

- a section of the world we want to reason about
- assertion
  - a sentence added to the knowledge about the domain
  - often uses the TELL construct
    - e.g. TELL `(KB-Fam, (Father(John) = Jim))`
  - sometimes ASSERT, RETRACT and MODIFY construct are used to make, withdraw and modify statements
- axiom
  - a statement with basic, factual, undisputed information about the domain
  - often used as definitions to specify predicates in terms of already defined predicates
- theorem
  - statement entailed by the axioms
  - it follows logically from the axioms

Example: Family Relationships

- objects: people
- properties: gender, …
  - expressed as unary predicates *Male(x), Female(y)*
- relations: parenthood, brotherhood, marriage
  - expressed through binary predicates *Parent(x,y), Brother(x,y), …*
- functions: motherhood, fatherhood
  - *Mother(x), Father(y)*
  - because every person has exactly one mother and one father
  - there may also be a relation *Mother-of(x,y), Father-of(x,y)*

Family Relationships

$\forall m,c\ Mother(c) = m \Leftrightarrow Female(m) \land Parent(m,c)$

$\forall w,h\ Husband(h,w) \Leftrightarrow Male(h) \land Spouse(h,w)$

$\forall x\ Male(x) \Leftrightarrow \neg Female(x)$

$\forall g,c\ Grandparent(g,c) \Leftrightarrow \exists p\ Parent(g,p) \land Parent(p,c)$

$\forall x,y\ Sibling(x,y) \Leftrightarrow \neg(x=y) \land \exists p\ Parent(p,x) \land Parent(p,y)$

. . .

Logic and the Wumpus World

- representation
  - suitability of logic to represent the critical aspects of the Wumpus World in a suitable way
- reflex agent
  - specification of a reflex agent for the Wumpus World
- change
  - dealing with aspects of the Wumpus World that change over time
- model-based agent
  - specification using logic

Reflex Agent in the Wumpus World

- rules that directly connect percepts to actions

  $\forall$ *b,g,u,c,t* *Percept([s, b, Glitter, u,c], t)* $\Rightarrow$ *Action(Grab, t)*

  - requires many rules for different combinations of percepts at different times
- can be simplified by intermediate predicates

  $\forall$ *s, b,g,u,c,t* *Percept([Stench, b, g, u, c], t)* $\Rightarrow$ *Stench(t)*

  $\forall$ *s, b,g,u,c,t* *Percept([s, Breeze, g, u, c], t)* $\Rightarrow$ *Breeze(t)*

  $\forall$ *s, b,g,u,c,t* *Percept([s, b, Glitter, u, c], t)* $\Rightarrow$ *AtGold(t)*

  $\forall$ *s, b,g,u,c,t* *Percept([s, b, g, Bump, c], t)* $\Rightarrow$ *Bump(t)*

  $\forall$ *s, b,g,u,c,t* *Percept([s, b, g, u, Scream], t)* $\Rightarrow$ *Scream(t)*

  $\forall$ *t* *AtGold(t)* $\Rightarrow$ *Action(Grab, t)*

  . . .

  - mainly abstraction over time
  - is it still a *reflex* agent?

Limitations of Reflex Agents

- the agent doesn't know its state
  - it doesn't know when to perform the climb action because it doesn't know if it has the gold, nor where the agent is
  - the agent may get into infinite loops because it will have to perform the same action for the same percepts

Change in the Wumpus World

- in principle, the percept history contains all the relevant knowledge for the agent
  - by writing rules that can access past percepts, the agent can take into account previous information
  - this is sufficient for optimal action under given circumstances
  - may be very tedious, involving many rules
- it is usually better to keep a set of sentences about the current state of the world
  - must be updated for every percept and every action

Agent Movement

- it is also helpful to provide constructs that help the agent keep track of its location, and how it can move
- essentially constructs a simple map for the agent
  - current location of the agent

    At(Agent, [1,1], S_0)

    uses a Situation parameter S_0 to keep track of changes independent of specific time points
  - orientation of the agent

    Orientation(Agent, S_0)
  - arrangement of locations, i.e. a map

    $\forall\ x, y$ *LocationToward([x,y],0) = [x+1,y]*

    $\forall\ x, y$ *LocationToward([x,y],90) = [x, y+1]*

    . . .

Model-Based Agent

- such an agent knows about locations through its map
  - it can associate properties with the locations
  - this can be used to reason about safe places, the presence of gold, pits, the wumpus, etc.

$\forall\, l,s \quad At(Agent,l,s) \wedge Breeze(s) \Rightarrow Breezy(l)$

. . .

$\forall\, l_1, l_2, s \quad At(Wumpus,l_1,s) \wedge Adjacent(l_1, l_2) \Rightarrow \text{Smelly}(l_2)$

. . .

$\forall\, l_1, l_2, s \quad \text{Smelly}(l_1) \Rightarrow (\exists\, l_2\ At(Wumpus,l_2,s)\ \neg(l_1 = l_2) \vee (Adjacent(l_1, l_2))$

. . .

$\forall\, l_1, l_2, x, t \quad \neg At(Wumpus, x,t) \wedge \neg\ (l_1 = l_2) \wedge \neg Pit(x)\ ) \Leftrightarrow OK(x)$

  - such an agent will find the gold provided there is a safe sequence
  - returning to the exit with the gold is difficult

Goal-Based Agent

- once the agent has the gold, it needs to return to the exit

  $\forall s$ Holding(Gold, s) $\Rightarrow$ GoalLocation([1,1],s)

- the agent can calculate a sequence of actions that will take it safely there
  - through inference
    - computationally rather expensive for larger worlds
    - difficult to distinguish good and bad solutions
  - through search
    - e.g. via the best-first search method
  - through planning
    - requires a special-purpose reasoning system

Utility-Based Agent

- can distinguish between more and less desirable states
  - different goals, pits, ...
    - pots with different amounts of gold
  - optimization of the route back to the exit
  - performance measure for the agent
  - requires the ability to deal with numbers in the knowledge representation scheme
    - possible in predicate logic, but tedious

Important Concepts and Terms

- agent
- `and`
- atomic sentence
- automated reasoning
- completeness
- conjunction
- constant
- disjunction
- domain
- existential quantifier
- fact
- `false`
- function
- implication
- inference mechanism
- inference rule
- interpretation
- knowledge representation
- logic
- model
- object
- `or`
- predicate
- predicate logic
- property
- proposition
- propositional logic
- propositional symbol
- quantifier
- query
- rational agent
- reflex agent
- relation
- resolution
- satisfiable sentence
- semantics
- sentence
- soundness
- syntax
- term
- `true`
- universal quantifier
- valid sentence
- variable

Chapter Summary

- logic can be used as the basis of formal knowledge representation and processing
  - syntax specifies the rules for constructing sentences
  - semantics establishes a relation between the sentences and their counterparts in the real world
  - simple sentences can be combined into more complex ones
  - new knowledge can be generated through inference rules from existing sentences
- propositional logic encodes knowledge about the world in simple sentences or formulae
- predicate logic is a formal language with constructs for the specifications of objects and their relations
  - models of reasonably complex worlds and agents can be constructed with predicate logic