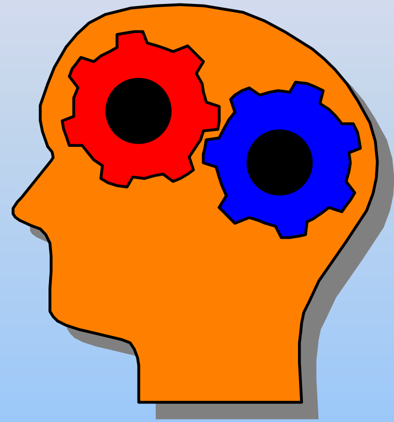
ARTIFICIAL INTELLIGENCE



CSC304

CZ3005

School of Computer Engineering
Nanyang Technological University

- I Artificial Intelligence
- II Problem Solving
- **III Knowledge and Reasoning**
- IV Acting Logically
- V Uncertain Knowledge and Reasoning
- VI Learning
- VII Communicating, Perceiving and Acting
- VIII Conclusions

©1996-2005 Michel Pasquier. All rights reserved

Profile of Lecturer

- **Dr Chai Quek**
- **Ph.D. – H.W. Edinburgh 1990**
  - An Intelligent Supervisory Control Schema
- **Area of Research: Learning Systems, Neural Network, Fuzzy System, Hybrid Fuzzy Neural Systems, Softcomputing, Computational Intelligence**
- **Application Areas: Computational Finance, Biomedical Engg, Intelligent Control, Intelligent Education, Soft modelling, (cognitive) sentiment mining**
- **Students groomed: over 12 gold medalists, 30 Ph.D.s, MSc, MEng etc.**
- **Hall 7 Head Counsellor, Assoc Chair (Students)**

©1996-2005 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - Knowledge-based Agents. – Representations.
  - Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - Syntax and Semantics. – Using First-Order Logic.
  - Logical Agents. – Representing Changes.
  - Deducing Properties of the World.
  - Goal-based Agents.
- **8 Building a Knowledge Base**
  - Knowledge Engineering. – General Ontology.

©1996-2005 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Example classes - Prolog as KBS**
  - Starting week 10 – intro
  - Starting week 12 – assignment Wk 14 (Venue TBA) **(marks to be part of continuous assessments)**

©1996-2005 Michel Pasquier. All rights reserved

6 – AGENTS THAT REASON LOGICALLY

"In which we design agents that can form representations of the world, use a process of inference to derive new representations about the world, and use these new representations to deduce what to do."

©1996-2005 Michel Pasquier. All rights reserved

The Knowledge-Based Approach

- **Agents that *know***
  - Achieve competence by being told new knowledge or by learning
  - Achieve adaptability by updating their knowledge
  - > *Knowledge representation*
    - State of the world, properties and evolution of the world; goals of the agent, actions and their effect
- **Agents that *reason***
  - Use knowledge to deduce course of actions
  - > *Knowledge inference*

Logic

©1996-2005 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - Knowledge-based Agents. – Representations.
  - Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - Syntax and Semantics. – Using First-Order Logic.
  - Logical Agents. – Representing Changes.
  - Deducing Properties of the World.
  - Goal-based Agents.
- **8 Building a Knowledge Base**
  - Knowledge Engineering. – General Ontology.

©1996-2005 Michel Pasquier. All rights reserved

Knowledge-Based Agents

- **Knowledge base (KB)**
  - Set of <u>sentences</u> i.e., representations of facts (DB)
  - Knowledge representation language
- **Adding and querying knowledge**
  - **Tell**: add a sentence to the KB
  - **Ask**: retrieve knowledge from the KB
    - Answers must *follow* from what has been **Tell**'ed (told)
- **Inference mechanism**
  - Role: determine what follows from the KB

©1996-2005 Michel Pasquier. All rights reserved

Problem Formulation of KBS

- **Knowledge Based System**
  - States: Instances of the KB (sets of sentences)

    –> Use **Tell** to build the KB

    e.g. Tell(KB, "Smoke ⇒ Fire")
    Tell(KB, "Fire ⇒ Call_911")
    …
    Tell(KB, "Smoke")
  - Operators: Add / Infer a new sentence
  - Goal: Answer a query

    –> Use **Ask** to query the KB

    e.g. Ask(KB, "**?** Call_911")

©1996-2005 Michel Pasquier. All rights reserved

A Generic Knowledge-Based Agent

```
function KB-Agent (percept) returns action
      static  KB,               // a knowledge base
              t                 // a time counter, initially 0

    Tell (KB, Make-Percept-Sentence (percept, t))
    action ← Ask (KB, Make-Action-Query (percept, t))
    Tell (KB, Make-Action-Sentence (action, t))
    t ← t + 1
    return action
```

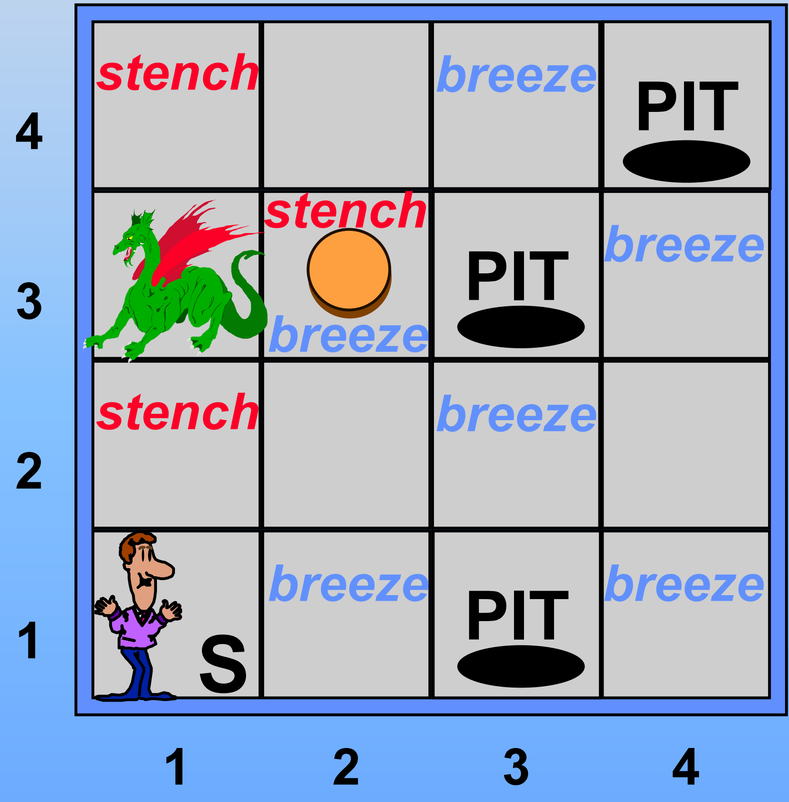
– > 3 steps: interpretation, inference, execution

– > KB: background knowledge (observed)
+ acquired information (deduced)

©1996-2005 Michel Pasquier. All rights reserved

Example: the Wumpus World

- **Problem description (PAGE)**
  - Environment
    - Grid of squares, walls;
    - Agent, gold, pits, wumpus.
  - Goal
    - Find the gold, return to S at [1,1].
  - Percepts
    - A list of 5 symbols, e.g. [Stench, Breeze, Glitter, Bump, Scream];
    - Agent's location *not* perceived.
  - Actions
    - Go-Forward, Turn-Left, Turn-Right, Grab, Shoot (1 arrow only), Climb.



©1996-2005 Michel Pasquier. All rights reserved

Levels of Knowledge

Tell Ask

- **Epistemological level**
  - <u>Declarative description</u> of knowledge
    - e.g. facts: “there is smoke in the kitchen”, “it is not warm enough”
      rules: “if there is smoke then there must be a fire”
- **Logical level**
  - <u>Logical encoding</u> of knowledge (into sentences)
    - e.g. facts: Smoke; rules: Implies(Smoke, Fire)
- **Implementation level**
  - <u>Physical representation</u> of knowledge (sentences)
    - e.g. - the string “Implies(Smoke, Fire)”, or
      - a “1” entry in a 2-dimensional array: Implies[X,Y]

©1996-2005 Michel Pasquier. All rights reserved

The Wumpus World

- **Problem description (cont'd)**
  - Initial state
    - Agent at [1,1]; gold, pits and wumpus in <u>random</u> squares.
  - Path-cost
    - Climbing out with the gold: +1000 (without: 0) • Each action: –1
    - Getting killed (pit or wumpus): –10000
  - <u>Knowledge</u>
    - "In all squares adjacent to the one where the wumpus is, the agent will perceive a <u>stench</u>."
    - "In all squares adjacent to a pit, the agent will perceive a <u>breeze</u>."
    - In the square where the gold is, the agent will perceive a <u>glitter</u>."
    - When walking into a wall, the agent will perceive a <u>bump</u>."
    - When the wumpus is killed, the agent will perceive a <u>scream</u>."

©1996-2005 Michel Pasquier. All rights reserved

Acting and Reasoning in the Wumpus World

(0) Initial state
[nil, nil, nil, nil, nil]

	1	2	3	4
4				
3				
2	*OK*			
1	A *OK*	*OK*		

(1) after {F}
[nil, Breeze, nil, nil, nil]

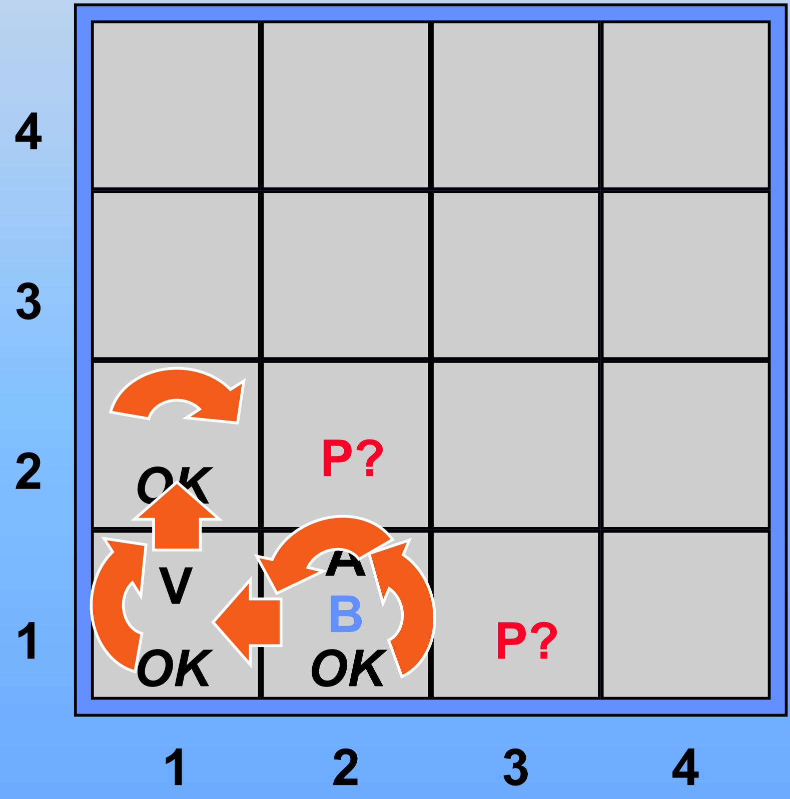
	1	2	3	4
4				
3				
2	*OK*	P?		
1	V *OK*	A B *OK*	P?	

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

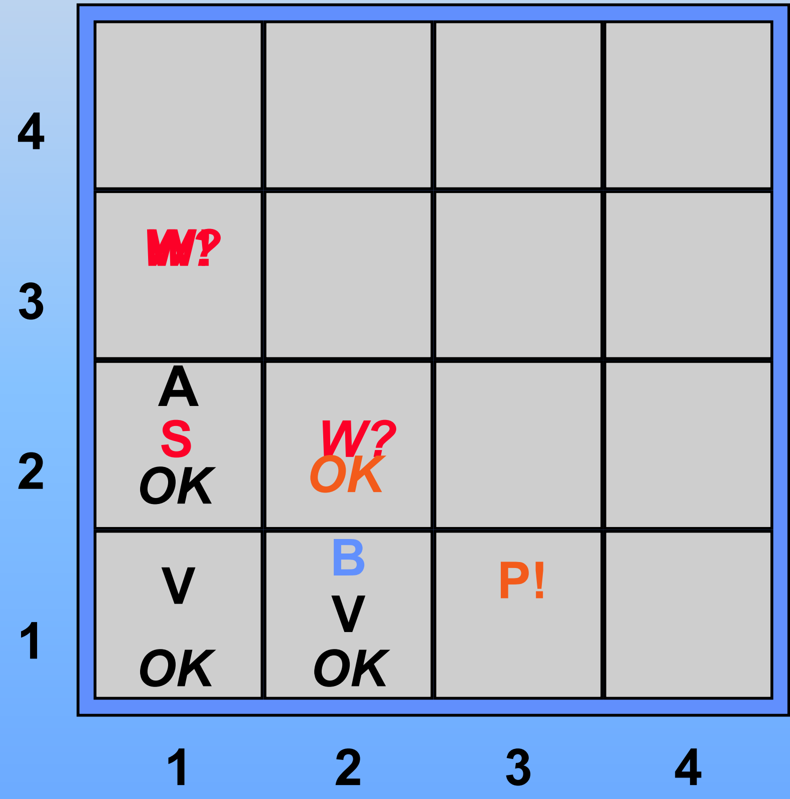
©1996-2005 Michel Pasquier. All rights reserved

Acting and Reasoning in the Wumpus World

(1) after {F}
[nil, Breeze, nil, nil, nil]



(6) after {F, L, L, F, R, F}
[Stench , nil, nil, nil, nil]



A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

©1996-2005 Michel Pasquier. All rights reserved

Acting and Reasoning in the Wumpus World

(6) after {F, L, L, F, R, F}
[Stench , nil, nil, nil, nil]

	1	2	3	4
4				
3	W!	OK		
2	S OK	OK	OK	
1	V OK	B V OK	P!	

(10) after {F, L, L, F, R, F ,R, F, L, F}
[Stench, Breeze, Glitter, nil, nil]

	1	2	3	4
4		P?		
3	W!	A S G B	P?	
2	S V OK	V OK	OK	
1	V OK	B V OK	P!	

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

©1996-2005 Michel Pasquier. All rights reserved

end

Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - – Knowledge-based Agents. – Representations.
  - – Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - – Syntax and Semantics. – Using First-Order Logic.
  - – Logical Agents. – Representing Changes.
  - – Deducing Properties of the World.
  - – Goal-based Agents.
- **8 Building a Knowledge Base**
  - – Knowledge Engineering. – General Ontology.

©1996-2005 Michel Pasquier. All rights reserved

Knowledge Representations

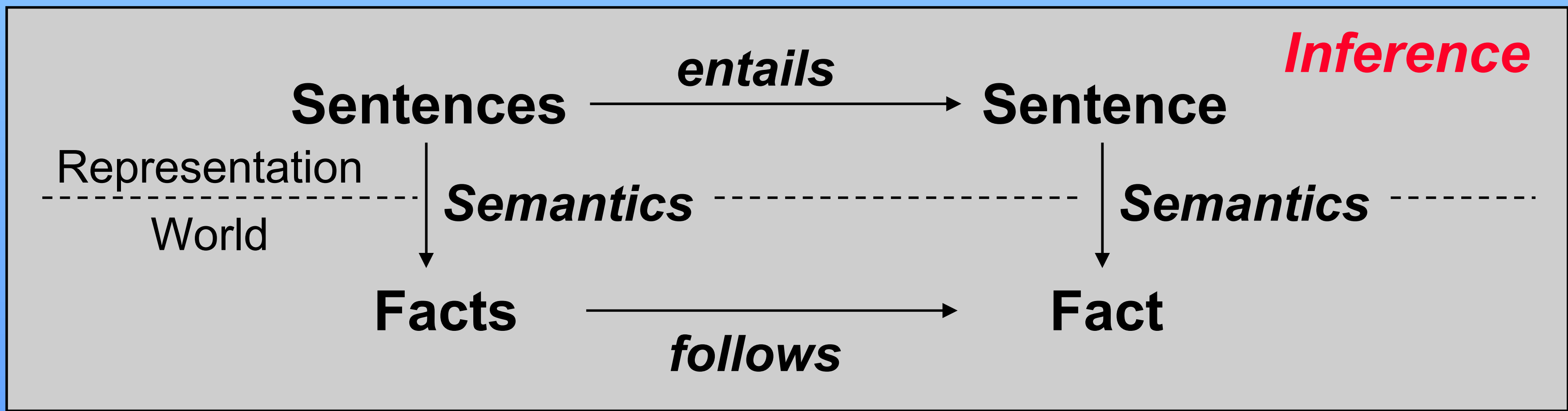
- **Knowledge representation (KR)**
  - KB: set of sentences –> need to
  - Express knowledge in a (computer-) tractable form
- **Knowledge representation language**
  - Syntax – implementation level
    - Possible configurations that constitute sentences
  - Semantics – knowledge level
    - Facts of the world the sentences refer to
    - e.g. language of arithmetics: x, y numbers
      sentence: “$x \geq y$”, semantics: “greater or equal”

Logic

©1996-2005 Michel Pasquier. All rights reserved

Reasoning and Logic

- **Logic**
  - Representation + Inference = Logic
    - Where representation = syntax + semantics
- **Reasoning**
  - Construction of new sentences from existing ones
- **Entailment as logical inference**



©1996-2005 Michel Pasquier. All rights reserved

Deduction and Induction

- **Mechanical reasoning**
  - Example
    - If a chord sequence is tonal, then it can be generated by a context-sensitive grammar.
    - The twelve-bar blues has a chord sequence that is tonal. |-
    - The twelve-bar blues has a chord sequence that can be generated by a context-sensitive grammar.
- **Deductive inference**
  - KB: Monday $\Rightarrow$ Work, Monday |- Work *sound (MP)*
- **Inductive inference**
  - KB: Monday $\Rightarrow$ Work, Work |- Monday *unsound!*
  - Generalization e.g., *"all swans are white …"*

©1996-2005 Michel Pasquier. All rights reserved

Entailment and Inference

- **Entailment**
  - Generate sentences that are necessarily true, given that the existing sentences are true
  - Notation: KB $\models \alpha$
    - e.g. Wumpus world:
      { "$\neg$S(1,1)", "$\neg$B(1,1)" } $\models$ "OK(2,1)"
      Arithmetics:
      { "$x \geq y$", "$y \geq z$" } $\models$ "$x \geq z$"
- **Inference**
  - **Tell**, given KB: (KB $\models \alpha$) !
  - **Ask**, given KB and α: (KB $\models \alpha$) ?

©1996-2005 Michel Pasquier. All rights reserved

Properties of Inference

- *Can be described by the sentences it derives, KB |=* α_I

- **Soundness**
  - Generate only entailed sentences
  - <u>Proof</u>: sequence of operations of a sound inference
    - Record of operations that generate a specific entailed sentence

      e.g. “Smoke ⇒ Fire” and “Smoke” |= “Fire”

      “Fire ⇒ Call_911” and “Fire” |= “Call_911”

- **Completeness**
  - A proof can be found for any entailed sentence

- **Proof theory**
  - Specify the reasoning operations that are sound

©1996-2005 Michel Pasquier. All rights reserved

An Example of Sound Inference

– Sentence: x

- Semantics: an expression; can be a single symbol or number, the concatenation of 2 expressions, etc.

– Sentence: x y

- Semantics: an expression which refers to a quantity that is the product of the quantities referred to by each of the expressions

– Sentence: x = y

- Semantics: the 2 expressions on each side of “**=**“ refer to the same quantity

– A sound inference: from

$$\begin{array}{l} E = mc^2 \\ T_1 \geq T_2 \end{array} \models E\,T_1 \geq mc^2\,T_2$$

©1996-2005 Michel Pasquier. All rights reserved

Knowledge Representation Languages

- **Formal (programming) languages**
  - Good at describing algorithms and data structures
    - e.g. the Wumpus world as a 4x4 array, World[2,2] ← Pit
  - Poor at representing incomplete / uncertain information
    - e.g. "there is a pit in [2,2] or [3,1]", or "…a wumpus *somewhere*"
  - > *not expressive enough*
- **Natural languages**
  - Very expressive (too much, thus very complex)
  - More appropriate for communication than representation
  - Suffer from ambiguity • e.g. "It's hot!"
    - e.g. "small cats and dogs" compared to "– x + y".

©1996-2005 Michel Pasquier. All rights reserved

Properties of Representations

- *KR languages should combine the advantages of both programming and natural languages.*

- **Desired properties**
  - Expressive
    - Can represent everything we need to.
  - Concise
  - Unambiguous
    - Sentences have a unique interpretation.
  - Context independent
    - Interpretation of sentences depends on semantics only.
  - Effective
    - An inference procedure allows to create new sentences.

©1996-2005 Michel Pasquier. All rights reserved

Properties of Semantics

- **Interpretation (meaning)**
  - *Correspondence between sentences and facts*
  - Arbitrary meaning, fixed by the writer of the sentence
    - e.g. Natural languages: meaning fixed by usage (cf. dictionary)
      exceptions: encrypted messages, codes (e.g. Morse)
      chat
  - Systematic relationship: compositional languages
    - *The meaning of a sentence is a function of the meaning of its parts.*
  - <u>Truth value</u>
    - A sentence make a claim about the world –> TRUE or FALSE
    - *Depends on the interpretation and the state of the world*
      - e.g. Wumpus world: S(1,2) true if means "Stench at [1,2]" and the world has a wumpus at either [1,3] or [2,2].

one
un
uno
yi
ichi
…
I
1
—
…

©1996-2005 Michel Pasquier. All rights reserved

Properties of Inference

- **Definition**
  - *Inference (reasoning) is the process by which conclusions are reached*
  - Logical inference (deduction) is the process that implements entailment between sentences
- **Useful properties**
  - Valid sentence (tautology)
    - iff TRUE under all possible interpretations in all possible worlds.
      - e.g. “S or ¬ S” is valid, “S(2,1) or ¬ S(2,1)”, etc.
  - Satisfiable sentence
    - iff there is some interpretation in some world for which it is TRUE
      - e.g. “S and ¬ S” is unsatisfiable

©1996-2005 Michel Pasquier. All rights reserved

Inference and Agent Programs

- **Inference in computers**
  - Does not know the interpretation the agent is using for the sentences in the KB
  - Does not know about the world (actual facts)
  - Knows only what appears in the KB (sentences)
    - e.g. Wumpus world: doesn't know the meaning of "OK", what a wumpus or a pit is, etc. – can only see: KB |= "[2,2] is OK"
  - > *Cannot reason informally*
    - does not matter, however, if KB |= "[2,2] is OK" is a valid sentence
- **Formal inference**
  - Can handle arbitrarily complex sentences, KB |= P

©1996-2005 Michel Pasquier. All rights reserved

Different Logics

- **Formal logic**
  - Syntax
    - A set of rules for writing sentences
  - Semantics
    - A set of rules (constraints) for relating sentences to facts
  - Proof theory / inference procedure
    - A set of rules for deducing entailments of sentences
- **Propositional logic**
  - Symbols, representing propositions (facts)
  - Boolean connectives, combining symbols
    - e.g. “Hot” or “Hot and Humid”

©1996-2005 Michel Pasquier. All rights reserved

Different Logics

- **First-order logic**
  - Objects and <u>predicates</u>, representing properties of and relations between objects
  - Variables, Boolean connectives and quantifiers
    - e.g. “Hot(x)”, “Hot(Air)” or ”Hot(Air) and Humid(Air)”
- **Temporal logic**
  - World ordered by a set of <u>time</u> points (intervals)
- **Probabilistic and fuzzy logic**
  - Degrees of <u>belief</u> and <u>truth</u> in sentences
    - e.g. “Washington is a state” with belief degree 0.4, “a city” 0.6, ”Washington is a large city” with truth degree 0.6

©1996-2005 Michel Pasquier. All rights reserved

Different Degrees of Truth

- *Q: Is there a tuna sandwich in the refrigerator?*
- *A: 0.5 !*

- **Probabilities**
  - There *is* or there *isn't* (50% chance either way).
- **Measures**
  - There is *half* a tuna sandwich there.
- **Fuzzy answer**
  - There is *something* there, but it *isn't really* a tuna sandwich. Perhaps it is some other kind of sandwich, or a tuna salad with no bread...

©1996-2005 Michel Pasquier. All rights reserved

The Commitments of Logics

Formal (KR) Language	**Ontological commitment** (what exists in the world)	**Epistemological commitment** (what an agent believes about facts)
Propositional logic	facts	true / false / unknown
First-order logic	facts, objects, relations	true / false / unknown
Temporal logic	facts, objects, rel., times	true / false / unknown
Probability logic	facts	degree of belief 0…1
Fuzzy logic	degrees of truth 0…1	degree of belief 0…1

©1996-2005 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - Knowledge-based Agents. – Representations.
  - Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - Syntax and Semantics. – Using First-Order Logic.
  - Logical Agents. – Representing Changes.
  - Deducing Properties of the World.
  - Goal-based Agents.
- **8 Building a Knowledge Base**
  - Knowledge Engineering. – General Ontology.

©1996-2005 Michel Pasquier. All rights reserved

Elements of Propositional Logic

- **Symbols**
  - Logical constants: TRUE, FALSE
  - Propositional symbols: P, Q, etc.
  - Logical connectives: $\wedge$, $\vee$, $\Leftrightarrow$, $\Rightarrow$, $\neg$
  - Parentheses: ( )
- **Sentences**
  - Atomic sentences: constants, propositional symbols
  - Combined with connectives, e.g. $P \wedge Q \vee R$
    also wrapped in parentheses, e.g. $(P \wedge Q) \vee R$

©1996-2005 Michel Pasquier. All rights reserved

Logical Connectives

- Conjunction $\wedge$
  - Binary op., e.g. $P \wedge Q$, “P and Q”, where P, Q are the *conjuncts*
- Disjunction $\vee$
  - Binary op., e.g. $P \vee Q$, “P or Q”, where P, Q are the *disjuncts*
- Implication $\Rightarrow$
  - Binary op., e.g. $P \Rightarrow Q$, “P implies Q”, where P is the *premise* (antecedent) and Q the *conclusion* (consequent)
  - Conditionals, “if-then” statements, or *rules*
- Equivalence $\Leftrightarrow$
  - Binary op., e.g. $P \Leftrightarrow Q$, “P equivalent to Q”
  - Biconditionals.
- Negation $\neg$
  - Unary op., e.g. $\neg P$, “not P”

©1996-2005 Michel Pasquier. All rights reserved

Syntax of Propositional Logic
(Backus-Naur Form)

Sentence	$\rightarrow$	AtomicSentence | ComplexSentence
AtomicSentence	$\rightarrow$	LogicalConstant | PropositionalSymbol
ComplexSentence	$\rightarrow$	(Sentence) | Sentence LogicalConnective Sentence | ¬Sentence
LogicalConstant	$\rightarrow$	TRUE | FALSE
PropositionalSymbol	$\rightarrow$	P | Q | R | …
LogicalConnective	$\rightarrow$	$\land$ | $\lor$ | $\Leftrightarrow$ | $\Rightarrow$ | $\neg$

Precedence (from highest to lowest): $\neg, \land, \lor, \Rightarrow, \Leftrightarrow$

e.g.: $\neg P \land Q \lor R \Rightarrow S$ (not ambiguous), eq. to: $(((\neg P) \land Q) \lor R) \Rightarrow S$

©1996-2005 Michel Pasquier. All rights reserved

Semantics of Propositional Logic

- **Interpretation of symbols**
  - Logical constants have fixed meaning
    - True: always means the fact is the case; valid
    - False: always means the fact is not the case; unsatisfiable
  - Propositional symbols mean “whatever they mean”
    - e.g.: **P** “we are in a pit”, etc.
    - Satisfiable, but not valid (true only when the fact is the case)
- **Interpretation of sentences**
  - Meaning derived from the meaning of its parts
    - Sentence as a combination of sentences using connectives
  - Logical connectives as (boolean) functions:

    TruthValue ***f*** (TruthValue, TruthValue)

©1996-2005 Michel Pasquier. All rights reserved

Semantics of Propositional Logic

- **Interpretation of connectives**
  - Truth-table
    - Define a mapping from input to output

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

  - Interpretation of sentences by decomposition
    - e.g.: $\neg P \land Q \lor R \Rightarrow S$, with $P \leftarrow T$, $Q \leftarrow T$, $R \leftarrow F$, $S \leftarrow F$:

$\neg P \leftarrow F$ $\quad$ $((\neg P) \land Q) \lor R) \leftarrow F$

$(\neg P) \land Q \leftarrow F$ $\quad$ $(((\neg P) \land Q) \lor R) \Rightarrow S \leftarrow T$

©1996-2005 Michel Pasquier. All rights reserved

end

Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - Knowledge-based Agents. – Representations.
  - Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - Syntax and Semantics. – Using First-Order Logic.
  - Logical Agents. – Representing Changes.
  - Deducing Properties of the World.
  - Goal-based Agents.
- **8 Building a Knowledge Base**
  - Knowledge Engineering. – General Ontology.

©1996-2005 Michel Pasquier. All rights reserved

Validity and Inference

- **Testing for validity**
  - Using truth-tables, checking all possible configurations
    - e.g.: $((P \vee Q) \wedge \neg Q) \Rightarrow P$

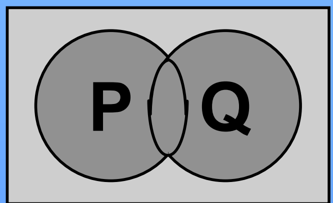
P	Q	$P \vee Q$	$\neg Q$	$(P \vee Q) \wedge \neg Q$	$((P \vee Q) \wedge \neg Q) \Rightarrow P$
False	False	False	True	False	True
False	True	True	False	False	True
True	False	True	True	True	True
True	True	True	False	False	True

- **A method for sound inference**
  - Build and check a truth-table for *Premises* $\Rightarrow$ *Conclusion*

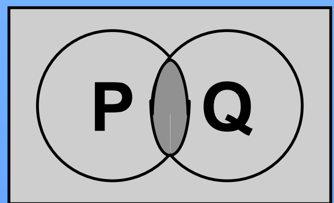
©1996-2005 Michel Pasquier. All rights reserved

Models

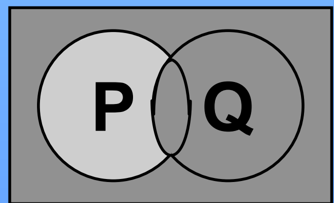
- **Definition**
  - *A world in which a sentence is true under a particular interpretation*
    - e.g. the Wumpus world example is a model for S(1,2) meaning “there is a stench in [1,2]”
  - Entailment: *KB* $|= \alpha$ *if the models of KB are all models of* α
- **Models as mathematical objects**
  - A mapping from propositional symbols to truth-values
- **Models as sets**



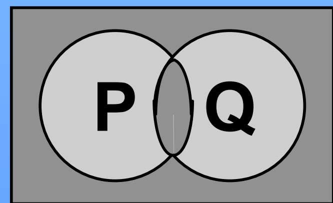
P $\vee$ Q



P $\wedge$ Q



P $\Rightarrow$ Q



P $\Leftrightarrow$ Q

©1996-2005 Michel Pasquier. All rights reserved

Rules of Inference

- **Sound inference rules**
  - Pattern of inference, that occur again and again
  - Soundness proven once and for all (truth-table)

$$\frac{\alpha}{\beta}$$

- **Classic rules of inference**
  - Implication-Elimination, or *Modus Ponens*
    - $$\frac{\alpha \Rightarrow \beta, \ \alpha}{\beta}$$

e.g. Cloudy $\wedge$ Humid $\Rightarrow$ Rain
Cloudy $\wedge$ Humid $\quad$ |= Rain

©1996-2005 Michel Pasquier. All rights reserved

Rules of Inference

- **Classic rules of inference**
  - And-Elimination
    - $$\frac{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}{\alpha_i}$$

    e.g. Cloudy $\land$ Humid $\models$ Cloudy
    Cloudy $\Rightarrow$ NoSunTan
  - And-Introduction
    - $$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}$$

    e.g. Cloudy, Humid
    Cloudy $\land$ Humid $\Rightarrow$ Rain
  - Or-Introduction
    - $$\frac{\alpha_i}{\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n}$$
  - Double-Negation-Elimination
    - $$\frac{\neg\neg\alpha}{\alpha}$$

©1996-2005 Michel Pasquier. All rights reserved

Rules of Inference

- **The resolution rule of inference**
  - Unit Resolution
    - $$\frac{\alpha \vee \beta, \quad \neg\beta}{\alpha}$$
  - Resolution
    - same as MP: $\dfrac{P \Rightarrow Q, \quad P}{Q}$ i.e. $\dfrac{\neg\beta \Rightarrow \alpha, \quad \neg\beta}{\alpha}$
    - $$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

e.g. monday $\vee$ tuesday, $\neg$ monday |= tuesday

Truth-table for the resolution

α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
False	False	False	False	True	False
False	False	True	False	True	True
False	True	False	True	False	False
False	True	True	True	True	True
True	False	False	True	True	True
True	False	True	True	True	True
True	True	False	True	False	True
True	True	True	True	True	True

©1996-2005 Michel Pasquier. All rights reserved

Equivalence Rules

- **Inference as implication**
  - Equivalent notations, e.g. MP:
    - $\dfrac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$ $\quad \alpha \Rightarrow \beta, \beta \ \vdash \ \beta \quad ((\alpha \Rightarrow \beta) \land \alpha) \Rightarrow \beta$

      $\alpha \Rightarrow \beta, \beta \ \models \ \beta$

- **Equivalence rules**
  - Associativity: $\alpha \land (\beta \land \gamma) \Leftrightarrow (\alpha \land \beta) \land \gamma$

    $\alpha \lor (\beta \lor \gamma) \Leftrightarrow (\alpha \lor \beta) \lor \gamma$
  - Distributivity: $\alpha \land (\beta \lor \gamma) \Leftrightarrow (\alpha \land \beta) \lor (\alpha \land \gamma)$

    $\alpha \lor (\beta \land \gamma) \Leftrightarrow (\alpha \lor \beta) \land (\alpha \lor \gamma)$
  - De Morgan's Law: $\neg(\alpha \lor \beta) \Leftrightarrow \neg\alpha \land \neg\beta$

    $\neg(\alpha \land \beta) \Leftrightarrow \neg\alpha \lor \neg\beta$

©1996-2005 Michel Pasquier. All rights reserved

Complexity of Inference

- **Proof by truth-table**
  - Complete
    - The truth-table can always be written.
  - Exponential time complexity
    - A proof involving ***N*** proposition symbols requires $\mathbf{2^N}$ rows.
    - In practice, a proof may refer only to a small subset of the KB.
- **Monotonicity**
  - Knowledge always increases

    if $KB_1 \models \alpha$ **then** $(KB_1 \cup KB_2) \models \alpha$
    - Allows for <u>local</u> rules,

      e.g. Modus Ponens $\alpha \Rightarrow \beta, \alpha \vdash \beta$
    - Propositional and first-order logic are monotonic.

©1996-2005 Michel Pasquier. All rights reserved

Horn Sentences

- **A particular sub-class of sentences**
  - Implication: $\mathbf{P_1 \wedge P_2 \wedge \ldots \wedge P_N \Rightarrow Q}$ where $P_1, \ldots P_N$, Q are non-negated atoms.
  - Particular cases:
    - $Q \Leftrightarrow (True \Rightarrow Q)$
    - $(P_1 \vee P_2 \vee \ldots \vee P_N \Rightarrow Q) \Leftrightarrow (P_1 \Rightarrow Q) \wedge \ldots \wedge (P_N \Rightarrow Q)$
    - $(P \Rightarrow Q_1 \wedge \ldots \wedge Q_N) \Leftrightarrow (P \Rightarrow Q_1) \wedge \ldots \wedge (P \Rightarrow Q_N)$
    - $(P \Rightarrow Q_1 \vee \ldots \vee Q_N)$ cannot be represented
- **Prolog, a logic programming language**
  - Horn sentences + Modus-Ponens $\mathbf{Q :\!- P_1, P_2, \ldots , P_N}$
  - Inference of <u>polynomial</u> time complexity

©1996-2005 Michel Pasquier. All rights reserved

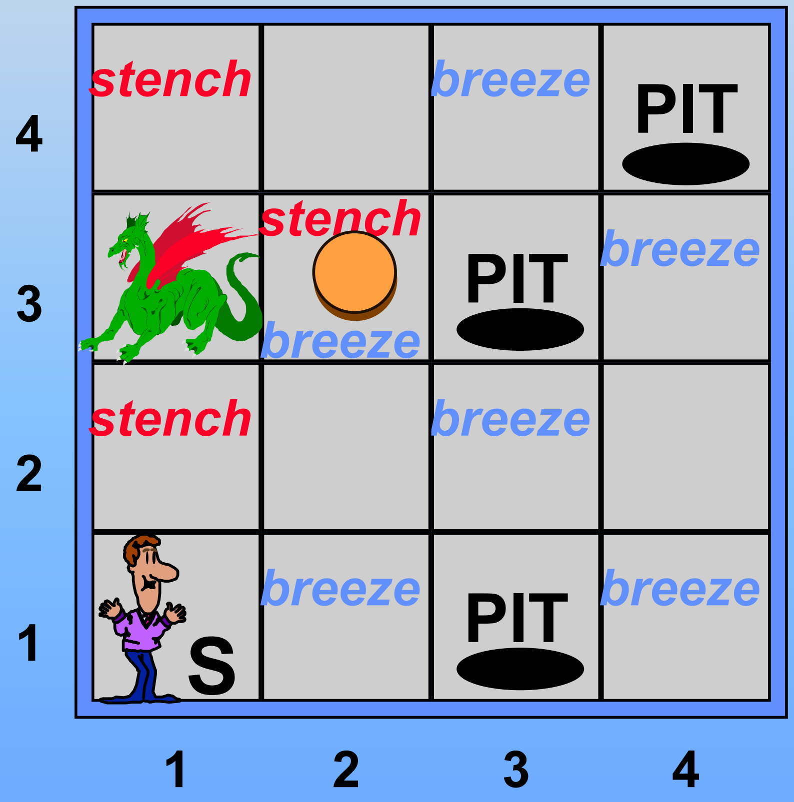
Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - Knowledge-based Agents. – Representations.
  - Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - Syntax and Semantics. – Using First-Order Logic.
  - Logical Agents. – Representing Changes.
  - Deducing Properties of the World.
  - Goal-based Agents.
- **8 Building a Knowledge Base**
  - Knowledge Engineering. – General Ontology.

©1996-2005 Michel Pasquier. All rights reserved

An Agent for the Wumpus World

- **A reasoning agent**
  - Propositional logic as the "programming language"
  - Knowledge base (KB) as problem representation
    - Percepts
    - Knowledge — sentences
    - Actions
  - Rule of inference (e.g. Modus Ponens) as the algorithm that will find a solution

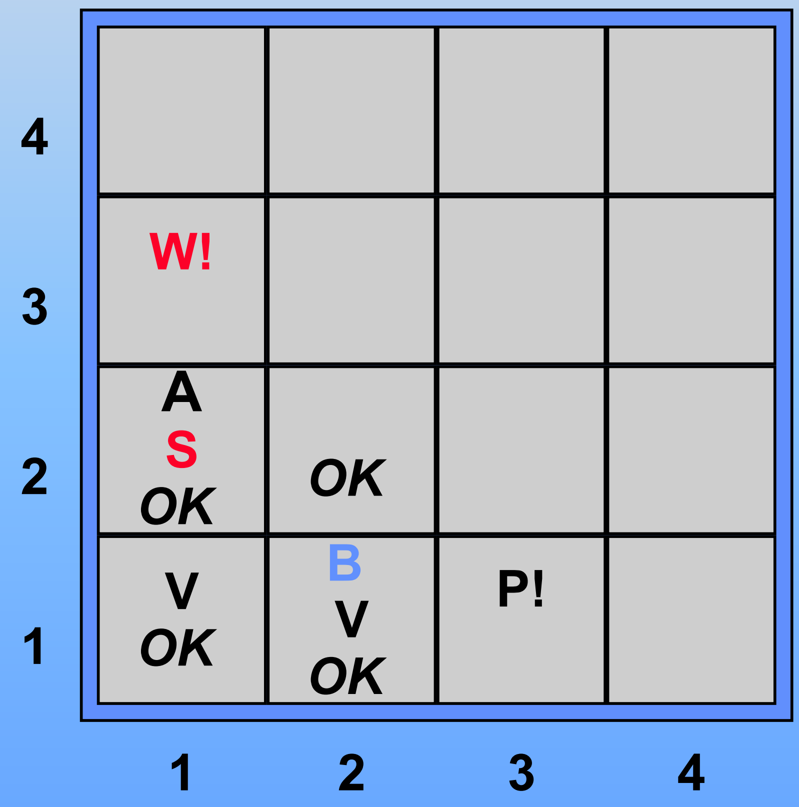


©1996-2005 Michel Pasquier. All rights reserved

The Knowledge Base

- **TELLing the KB: percepts**
  - Syntax and semantics
    - Symbol **S11**, meaning "there is a stench at [1,1]"
    - Symbol **B12**, meaning "there is a breeze at [1,2]"
      …
- **Percept sentences**
  - Partial list:
    - ¬ S11, ¬ B11, ¬ G11, …
      ¬ S21, B21, ¬ G21, …
      S12, ¬ B12, ¬ G11, …
      …

[Stench , nil, nil, nil, nil]



©1996-2005 Michel Pasquier. All rights reserved

The Knowledge Base

- **TELLing the KB: knowledge**
  - Rules about the environment
    - "All squares adjacent to the wumpus have a stench."

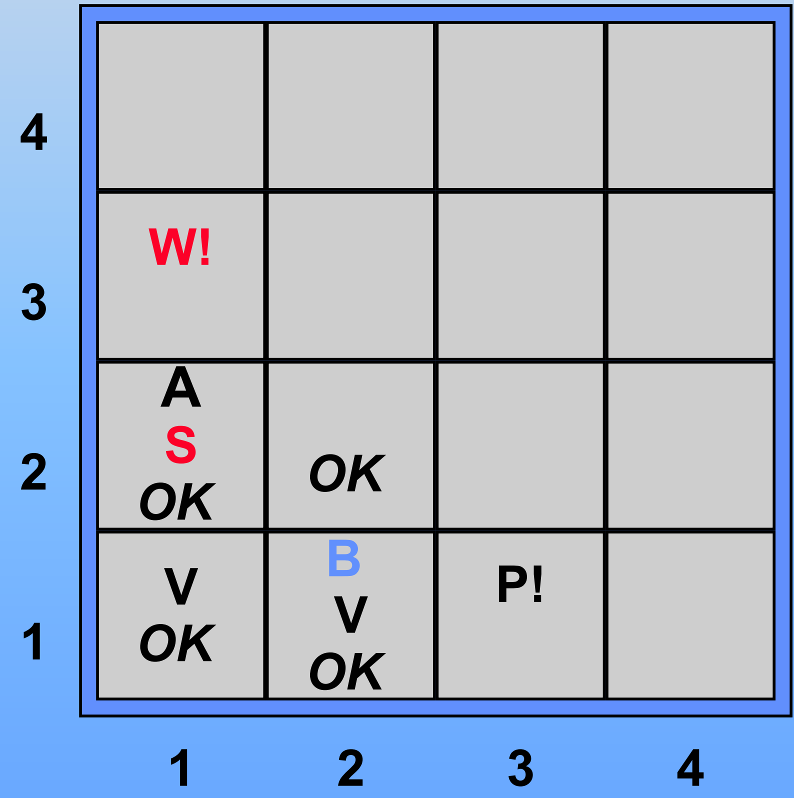
      $S12 \Rightarrow W11 \vee W12 \vee W22 \vee W13$
    - "A square with no stench has no wumpus and adjacent squares have no wumpus either."

      $\neg S11 \Rightarrow \neg W11 \wedge \neg W21 \wedge \neg W12$

      $\neg S21 \Rightarrow \neg W11 \wedge \neg W21 \wedge \neg W22 \wedge \neg W31$

      $\neg S12 \Rightarrow \neg W11 \wedge \neg W12 \wedge \neg W22 \wedge \neg W13$

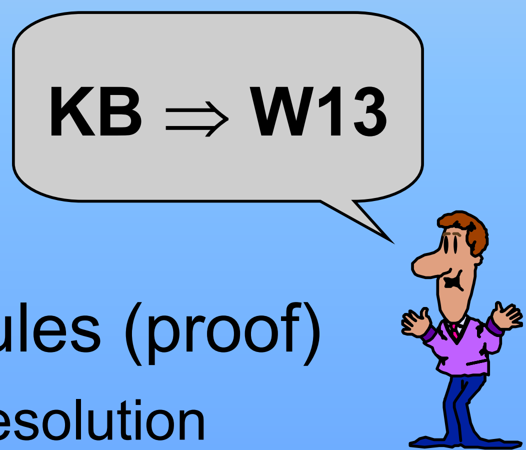
[Stench , nil, nil, nil, nil]



©1996-2005 Michel Pasquier. All rights reserved

Finding the Wumpus

- **Checking the truth-table**
  - Exhaustive check: every row for which KB is true also has W13 true
    - 12 propositional symbols, i.e.
      S11, S21, S12, W11, W21, W12, W22, W13, W31, B11, B21, B12
    - $2^{12} = 4096$ rows
  - > *possible, but lengthy* *impossible for the complete problem*



- **Reasoning by inference**
  - Application of a sequence of inference rules (proof)
    - Modus Ponens, And-Elimination, and Unit-Resolution

©1996-2005 Michel Pasquier. All rights reserved

Proof for “KB $\Rightarrow$ W13”

Knowledge Base

$\neg$ S11,	$\neg$ B11,	$\neg$ G11,
$\neg$ S21,	B21,	$\neg$ G21,
S12,	$\neg$ B12,	$\neg$ G11,

R1: $\neg S11 \Rightarrow \neg W11 \land \neg W21 \land \neg W12$

R2: $\neg S21 \Rightarrow \neg W11 \land \neg W21 \land \neg W22 \land \neg W31$

R3: $\neg S12 \Rightarrow \neg W11 \land \neg W12 \land \neg W22 \land \neg W13$

R4: $S12 \Rightarrow W11 \lor W12 \lor W22 \lor W13$

Inferences

1. Modus Ponens: $\neg$ S11, **R1**
   $\vdash$ $\neg W11 \land \neg W21 \land \neg W12$
2. And-Elimination: ♦
   $\vdash$ $\neg W11$, $\neg W21$, $\neg W12$
3. Modus Ponens: $\neg$ S21, **R2**
   $\vdash$ $\neg W11 \land \neg W21 \land \neg W22 \land \neg W31$
4. And-Elimination: ♦
   $\vdash$ $\neg W11$, $\neg W21$, $\neg W22$, $\neg W31$

©1996-2005 Michel Pasquier. All rights reserved

Proof for "KB $\Rightarrow$ W13"

Knowledge Base

$\neg S11, \quad \neg B11, \quad \neg G11,$
$\neg S21, \quad B21, \quad \neg G21,$
$S12, \quad \neg B12, \quad \neg G11,$

R1: $\neg S11 \Rightarrow \neg W11 \wedge \neg W21 \wedge \neg W12$

R2: $\neg S21 \Rightarrow \neg W11 \wedge \neg W21 \wedge \neg W22 \wedge \neg W31$

R3: $\neg S12 \Rightarrow \neg W11 \wedge \neg W12 \wedge \neg W22 \wedge \neg W13$

R4: $S12 \Rightarrow W11 \vee W12 \vee W22 \vee W13$

Inferences

KB += $\neg W11, \neg W21, \neg W12, \neg W22, \neg W31$

5. Modus Ponens: S12, **R4**
   $\vdash \quad (W13 \vee W12 \vee W22) \vee W11$

6. Unit-Resolution: ♦, $\neg W11$
   $\vdash \quad (W13 \vee W12) \vee W22$

©1996-2005 Michel Pasquier. All rights reserved

Proof for “KB $\Rightarrow$ W13”

Knowledge Base

$\neg$ S11, $\neg$ B11, $\neg$ G11,
$\neg$ S21, B21, $\neg$ G21,
S12, $\neg$ B12, $\neg$ G11,

R1: $\neg S11 \Rightarrow \neg W11 \land \neg W21 \land \neg W12$

R2: $\neg S21 \Rightarrow \neg W11 \land \neg W21 \land \neg W22 \land \neg W31$

R3: $\neg S12 \Rightarrow \neg W11 \land \neg W12 \land \neg W22 \land \neg W13$

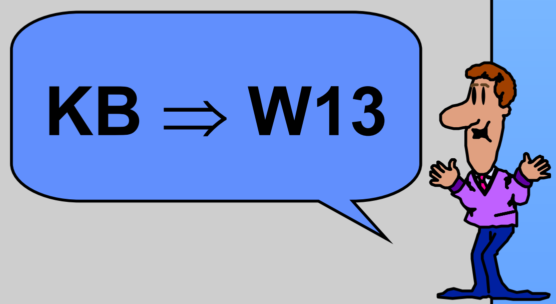
R4: $S12 \Rightarrow W11 \lor W12 \lor W22 \lor W13$

Inferences

KB += $\neg W11$, $\neg W21$, $\neg W12$, $\neg W22$, $\neg W31$, $(W13 \lor W12) \lor W22$

7. Unit-Resolution: ♦, $\neg W22$
   $\vdash$ $W13 \lor W12$

8. Unit-Resolution: ♦, $\neg W12$
   $\vdash$ W13



©1996-2005 Michel Pasquier. All rights reserved

From Knowledge to Actions

- **TELLing the KB: actions**
  - Additional rules
    - e.g. “if the wumpus is 1 square ahead then do not go forward”

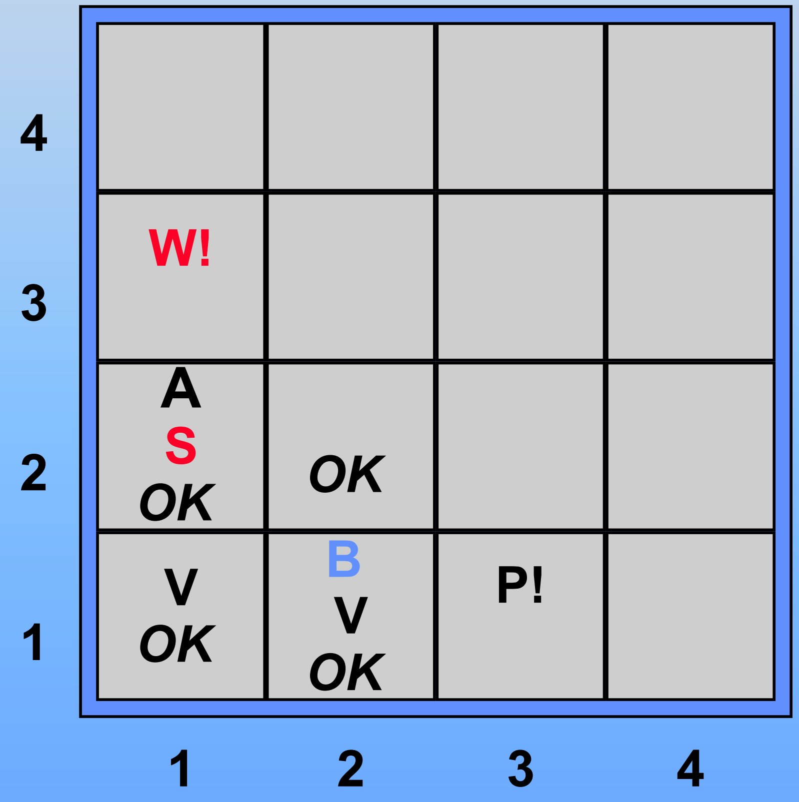
      A12 ∧ East ∧ W22 ⇒ ¬Forward

      A12 ∧ North ∧ W13 ⇒ ¬Forward

      …

- **ASKing the KB**
  - Cannot ask “which action?” but “should I go forward?”

[Stench , nil, nil, nil, nil]



©1996-2005 Michel Pasquier. All rights reserved

A Knowledge-Based Agent Using Propositional Logic

```
function Propositional-KB-Agent (percept) returns action
        static  KB,                 // a knowledge base
                t                   // a time counter, initially 0

        Tell (KB, Make-Percept-Sentence (percept, t))
        foreach action in the list of possible actions
        do
           if Ask (KB, Make-Action-Query (t, action)) then
                Tell (KB, Make-Action-Sentence (action, t))
                t ← t + 1
                return action
        end
```


©1996-2005 Michel Pasquier. All rights reserved

The Limits of Propositional Logic

- **A weak logic**
  - Too many propositions to TELL the KB
    - e.g. the rule “if the wumpus is 1 square ahead then do not go forward” needs 64 sentences (16 squares x 4 orientations)!
    - Result in increased time complexity of inference
  - Handling change is difficult
    - Need time-dependent propositional symbols
      e.g. A11 means “the agent is in square [1,1]” - when?
      at t = 0: A11-0; at t = 1: A21-1;
      at t = 2: A11-2
    - Need to rewrite rules as time-dependent
      e.g. A12-0 $\wedge$ East-0 $\wedge$ W22-0 $\Rightarrow$ $\neg$Forward-0
      A12-2 $\wedge$ East-2 $\wedge$ W22-2 $\Rightarrow$ $\neg$Forward-2

©1996-2005 Michel Pasquier. All rights reserved

Summary

- **Intelligent agents need …**
  - Knowledge about the world, so as to take good decisions.
- **Knowledge can be …**
  - Defined using a knowledge representation language.
  - Stored in a knowledge base in the form of sentences.
  - Inferred, using an inference mechanism and rules.
- **A representation language is defined by …**
  - A syntax, which specify the structure of sentences, and
  - A semantics, which specifies how the sentences relate to facts in the world.

©1996-2005 Michel Pasquier. All rights reserved

Summary

- **Inference is …**
  - The process of deducing new sentences from old ones.
  - Sound if it derives true conclusions from true premises.
  - Complete if it can derive all possible true conclusions.
- **Logics …**
  - Make different commitments about what the world is made of and what kind of beliefs we can have about facts.
  - Are useful for the commitments they *do not* make.
- **Propositional logic …**
  - Commits only to the existence of facts.
  - Has simple syntax and semantics and is therefore limited.

©1996-2005 Michel Pasquier. All rights reserved

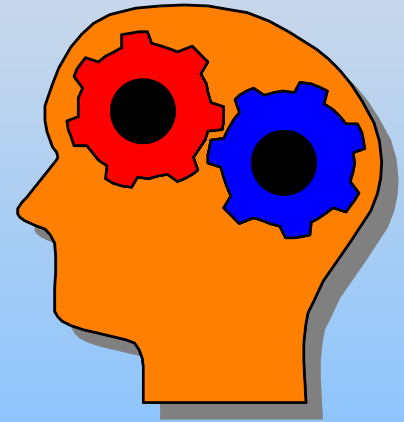
References

- Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18(1):82-127.
- Quine, W. V. (1960). *Word and Object*. MIT Press, Cambridge, Massachusetts.
- Rosenschein, S. J. (1985). Formal theories of knowledge in AI and robotics. *New Generation Computing*, 3(4):345-357.
- Tarski, A. (1956). *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*. Oxford University Press, Oxford.

©1996-2005 Michel Pasquier. All rights reserved

end

ARTIFICIAL INTELLIGENCE



CSC304
CPE406
SC430

School of Computer Engineering
Nanyang Technological University

Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - Knowledge-based Agents. – Representations.
  - Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - Syntax and Semantics. – Using First-Order Logic.
  - Logical Agents. – Representing Changes.
  - Deducing Properties of the World.
  - Goal-based Agents.
- **8 Building a Knowledge Base**
  - Knowledge Engineering. – General Ontology.

©1996-2006 Michel Pasquier. All rights reserved

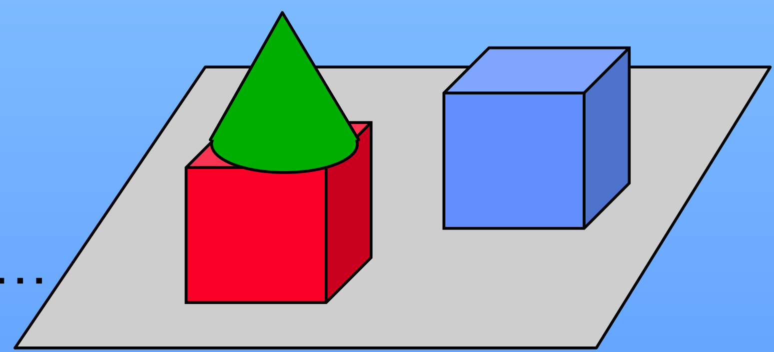
7 – FIRST-ORDER LOGIC

"In which we introduce a logic that is sufficient for building knowledge-based agents."

©1996-2006 Michel Pasquier. All rights reserved

Representing Knowledge

- **Knowledge-based agent**
  - Have representations of the world in which they operate
  - Use those representations to infer what actions to take
- **Ontological commitments**
  - The world as facts (propositional logic)
  - The world as objects (first-order logic) with properties *about* each object, and relations *between* objects
    - e.g. the blocks world:
      Objects: cubes, cylinders, cones, …
      Properties: shape, colour, location, …
      Relations: above, under, next-to, …



©1996-2006 Michel Pasquier. All rights reserved

Why is FOL Important?

- **A very powerful KR scheme**
  - Essential representation of the world
    - Deal with objects, properties, and relations (as Philosophy).
  - Simple, generic representation
    - Does <u>not</u> deal with specialised concepts such as categories, time, and events.
  - Universal language
    - Can express anything that can be programmed.
  - Most studied and best understood
    - More powerful proposals still debated.
    - Less powerful schemes too limited.

©1996-2006 Michel Pasquier. All rights reserved

Propositional vs. First-Order Logic

- **Aristotle's syllogism**
  - Socrates is a man. All men are mortal. Therefore Socrates is mortal.

Statement	Propositional Logic	First-Order Logic
"Socrates is a man."	SocratesMan, S43	Man(Socrates), P52(S21)
"Plato is a man."	PlatoMan, S157	Man(Plato), P52(S99)
"All men are mortal."	MortalMan S421 Man $\Rightarrow$ Mortal S9$\Rightarrow$S4	Man(x) $\Rightarrow$ Mortal(x), P52(V1) $\Rightarrow$ P66(V1)
"Socrates is mortal."	MortalSocrates S957 S43 $\wedge$ S421 $\vdash$ S957 ?!?	Mortal(Socrates) V1$\leftarrow$S21, … $\vdash$ P66(S21)

©1996-2006 Michel Pasquier. All rights reserved

Syntax and Semantics of FOL

- **Sentences**
  - Built from quantifiers, predicate symbols, and terms
- **Terms**
  - Represent objects
  - Built from variables, constant and function symbols
- **Constant symbols**
  - Refer to ("name") particular objects of the world
    - The object is specified by the interpretation
      e.g. "John" is a constant , may refer to "John, king of England from 1199 to 1216 and younger brother of Richard Lionheart", or my uncle, or …

©1996-2006 Michel Pasquier. All rights reserved

Predicate and Function Symbols

- **Predicate symbols**
  - Refer to particular relations on objects
    - Binary relation specified by the interpretation
      e.g. Brother(KingJohn, RichardLionheart) –> T or F
  - A *n-ary* relation if defined by a set of *n-tuples*
    - Collection of objects arranged in a fixed order
      e.g. { ⟨KingJohn, RichardLionheart⟩, ⟨KingJohn, Henry⟩, ... }
- **Function symbols**
  - Refer to functional relations on objects
    - Many-to-one relation specified by the interpretation
      e.g. BrotherOf(KingJohn) –> a person, e.g. Richard (*not* T/F)
  - Defined by a set of *n+1-tuples*
    - Last element is the function value for the first *n* elements.

©1996-2006 Michel Pasquier. All rights reserved

Variables and Terms in FOL

- **Variables**
  - Refer to any object of the world
    - e.g. x, person, … as in Brother(KingJohn, person).
  - Can be substituted by a constant symbol
    - e.g. person ← Richard, yielding Brother(KingJohn, Richard).
- **Terms**
  - Logical expressions referring to objects
    - Include constant symbols (“names”) and variables.
    - Make use of function symbols.
      e.g. LeftLegOf(KingJohn) to refer to his leg without naming it
  - Compositional interpretation
    - e.g. LeftLegOf(), KingJohn –> LeftLegOf(KingJohn).

©1996-2006 Michel Pasquier. All rights reserved

Sentences in FOL

- **Atomic sentences**
  - State facts, using terms and predicate symbols
    - e.g. Brother(Richard, John).
  - Can have complex terms as arguments
    - e.g. Married(FatherOf(Richard), MotherOf(John)).
  - Have a truth value
    - Depends on both the interpretation and the world.
- **Complex sentences**
  - Combine sentences with connectives
    - e.g. Father(Henry, KingJohn) $\wedge$ Mother(Mary, KingJohn)
  - Connectives identical to propositional logic
    - i.e.: $\wedge$, $\vee$, $\Leftrightarrow$, $\Rightarrow$, $\neg$

©1996-2006 Michel Pasquier. All rights reserved

Sentence Equivalence

- ***There are many ways to write a logical statement in FOL***
  - Example
    - $A \Rightarrow B$ equivalent to $\neg A \vee B$

      "rule form" *"complementary cases"*

      $Dog(x) \Rightarrow Mammal(x)$ $\neg Dog(x) \vee Mammal(x)$

      "dogs are mammals" "either it's not a dog or it's a mammal"
    - $A \wedge B \Rightarrow C$ equivalent to $A \Rightarrow (B \Rightarrow C)$

Proof: $A \wedge B \Rightarrow C \Leftrightarrow \neg (A \wedge B) \vee C \Leftrightarrow (\neg A \vee \neg B) \vee C$

$\Leftrightarrow \boldsymbol{\neg A \vee \neg B \vee C} \Leftrightarrow \neg A \vee (\neg B \vee C)$

$\Leftrightarrow \neg A \vee (B \Rightarrow C) \Leftrightarrow A \Rightarrow (B \Rightarrow C)$

$\boldsymbol{\neg P \vee Q \Leftrightarrow P \Rightarrow Q}$

©1996-2006 Michel Pasquier. All rights reserved

Sentences in Normal Form

- ***There is only one way to write a logical statement using a Normal Form of FOL***
  - Example $\neg B \Rightarrow \neg A$
    - $A \Rightarrow B$, $A \wedge B \Rightarrow C$ equivalent to $\neg A \vee B$, $\neg A \vee \neg B \vee C$
      "Implicative Normal Form" "Conjunctive Normal Form"
- ***Rewriting logical sentences allows to determine whether they are equivalent or not***
  - Example
    - $A \wedge B \Rightarrow C$ and $A \Rightarrow (B \Rightarrow C)$
      both have the same CNF: $\neg A \vee \neg B \vee C$
- ***Using FOL is the most convenient, but using a Normal Form is the most efficient***

©1996-2006 Michel Pasquier. All rights reserved

Sentence Verification

- ***Rewriting logical sentences helps to understand their meaning***
  - Example
    - $\text{Owns}(x,y) \Rightarrow (\text{Dog}(y) \Rightarrow \text{AnimalLover}(x))$ $\quad A \Rightarrow (B \Rightarrow C)$

      $\text{Owns}(x,y) \wedge \text{Dog}(y) \Rightarrow \text{AnimalLover}(x)$ $\quad A \wedge B \Rightarrow C$

      "A dog owner is an animal lover"
- ***Rewriting logical sentences helps to verify their meaning is as intended***
  - Example
    - *"Dogs all have the same enemies"*

      $\text{Dog}(x) \wedge \text{Enemy}(z, x) \Rightarrow (\text{Dog}(y) \Rightarrow \text{Enemy}(z, y))$ same as

      $\text{Dog}(x) \wedge \text{Dog}(y) \wedge \text{Enemy}(z, x) \Rightarrow \text{Enemy}(z, y)$

©1996-2006 Michel Pasquier. All rights reserved

Universal Quantifier $\forall$

- **Express properties of collections of objects**
  - Make a statement about *every* objects w/out enumerating
    - e.g. “All kings are mortal

      King(Henry) $\Rightarrow$ Mortal(Henry) Λ
      King(John) $\Rightarrow$ Mortal(John) Λ
      King(Richard) $\Rightarrow$ Mortal(Richard) Λ
      King(London) $\Rightarrow$ Mortal(London) Λ
      …

      instead: $\forall$ x, King(x) $\Rightarrow$ Mortal(x)

    - Note: the semantics of the implication says F $\Rightarrow$ F is TRUE,
      thus for those individuals that satisfy the premise King(x) the rule asserts the conclusion Mortal(x)
      but for those individuals that do not satisfy the premise the rule makes no assertion.

©1996-2006 Michel Pasquier. All rights reserved

Using the Universal Quantifier

- ***The implication ($\Rightarrow$) is the natural connective to use with the universal quantifier ($\forall$)***
  - Example
    - General form: $\forall x\ P(x) \Rightarrow Q(x)$ e.g. $\forall x\ Dog(x) \Rightarrow Mammal(x)$
      "all dogs are mammals"
    - Use conjunction? $\forall x\ P(x) \land Q(x)$ e.g. $\forall x\ Dog(x) \land Mammal(x)$

      same as $\forall x\ P(x)$ and $\forall x\ Q(x)$
      e.g. $\forall x\ Dog(x)$ and $\forall x\ Mammal(x)$

      –> yields a very strong statement (too strong! i.e. *incorrect*)

©1996-2006 Michel Pasquier. All rights reserved

Existential Quantifier $\exists$

- **Express properties of some particular objects**
  - Make a statement about *one* object without naming it
    - e.g. “King John has a brother who is king”

      $\exists$ x, Brother(x, KingJohn) $\wedge$ King(x)

    instead of

      Brother(Henry, KingJohn) $\wedge$ King(Henry) $\vee$
      Brother(KingJohn, KingJohn) $\wedge$ King(KingJohn) $\vee$
      Brother(Mary, KingJohn) $\wedge$ King(Mary) $\vee$
      Brother(London, KingJohn) $\wedge$ King(London) $\vee$
      Brother(Richard, KingJohn) $\wedge$ King(Richard) $\vee$
      …

©1996-2006 Michel Pasquier. All rights reserved

Using the Existential Quantifier

- ***The conjunction (Λ) is the natural connective to use with the existential quantifier ($\exists$)***
  - Example
    - General form: $\exists x\ P(x) \land Q(x)$ e.g. $\exists x\ Dog(x) \land Owns(John, x)$
      "John owns a dog"
    - Use Implication? $\exists x\ P(x) \Rightarrow Q(x)$ e.g. $\exists x\ Dog(x) \Rightarrow Owns(John, x)$

      true for all x such that P(x) is false
      e.g. $Dog(Garfield) \Rightarrow Owns(John, Garfield)$

      –> yields a very weak statement (too weak! i.e. *useless*)

©1996-2006 Michel Pasquier. All rights reserved

end

Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - Knowledge-based Agents. – Representations.
  - Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - Syntax and Semantics. – Using First-Order Logic.
  - Logical Agents. – Representing Changes.
  - Deducing Properties of the World.
  - Goal-based Agents.
- **8 Building a Knowledge Base**
  - Knowledge Engineering. – General Ontology.

©1996-2006 Michel Pasquier. All rights reserved

Nesting and Mixing Quantifiers

- **Combining $\forall$ and $\exists$**
  - Express more complex sentences
    - e.g. “if x is the parent of y then y is the child of x”:
      $\forall x, \forall y \ \ Parent(x, y) \Rightarrow Child(y, x)$
      ”every person has a parent”: $\forall x \ Person(x) \Rightarrow \exists y \ Parent(y, x)$
  - Semantics depends on quantifiers ordering
    - e.g. $\exists y, \forall x \ \ Parent(y, x)$
      ”there is someone who is everybody’s parent” ?!?
  - Choosing variables to avoid confusion
    - e.g. $\forall x \ King(x) \vee \exists x \ Brother(Richard, x)$ is better written:
      $\forall x \ King(x) \vee \exists z \ Brother(Richard, z)$
- **Well-formed formula (WFF)**
  - Sentences with all variables properly quantified

©1996-2006 Michel Pasquier. All rights reserved

Connections between Quantifiers

- **Equivalences**
  - Using the negation *(hence only one quantifier is needed)*

    $$\forall x \; P(x) \Leftrightarrow \neg \exists x \neg P(x)$$

    - e.g. "everyone is mortal":

      $$\forall x \; Mortal(x) \Leftrightarrow \neg \exists x \neg Mortal(x)$$

  - De Morgan's Laws
$\forall x\, P \Leftrightarrow \neg \exists x\, \neg P$	$P \wedge Q \Leftrightarrow \neg(\neg P \vee \neg Q)$
$\forall x\, \neg P \Leftrightarrow \neg \exists x\, P$	$\neg P \wedge \neg Q \Leftrightarrow \neg(P \vee Q)$
$\neg \forall x\, P \Leftrightarrow \exists x\, \neg P$	$\neg (P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$
$\neg \forall x\, \neg P \Leftrightarrow \exists x\, P$	$\neg(\neg P \wedge \neg Q) \Leftrightarrow P \vee Q$

©1996-2006 Michel Pasquier. All rights reserved

Equality Predicate Symbol

- **Need for equality**
  - State that two terms refer to the <u>same object</u>
    - e.g. Father(John) **=** Henry, or **=**(Father(John), Henry)
  - Predicate symbol with fixed semantics
    - <u>Identity relation</u>, i.e. the set of pairs (2-tuples) of objects where both elements of a pair are the same object.

      e.g. { ⟨Henry, Henry⟩,
      ⟨KingJohn, KingJohn⟩,
      ⟨RichardLionheart, RichardLionheart⟩,
      … }
  - Useful to define properties
    - e.g. “King John has two brothers”:

      $\exists x,y$ Brother(x, KingJohn) $\wedge$ Brother(y, KingJohn) $\wedge$ $\neg(x=y)$

©1996-2006 Michel Pasquier. All rights reserved

Grammar of First-Order Logic
(Backus-Naur Form)

Sentence	→	AtomicSentence | (Sentence) | Sentence Connective Sentence | ¬Sentence | Quantifier Variable, … Sentence
AtomicSentence	→	Predicate(Term, …) | Term = Term
Term	→	Function(Term, …) | Constant | Variable
Connective	→	$\wedge$ | $\vee$ | $\Leftrightarrow$ | $\Rightarrow$
Quantifier	→	$\forall$ | $\exists$
Constant	→	A | X_1 | John | …
Variable	→	a | x | person | …
Predicate	→	P() | Colour() | Before() | …
Function	→	F() | MotherOf() | SquareRootOf() | …

©1996-2006 Michel Pasquier. All rights reserved

Extensions to First-Order Logic

- **Higher-order logics**
  - First-order logic: quantifiers over objects
    - e.g. $\forall x,y\ \ Equal(x, y) \Leftrightarrow \forall x,y\ (x=y)$
  - Second-order logic: quantifiers over relations
    - e.g. "2 objects are equal iff all properties are equivalent":
      $\forall x,y\ \ Equal(x, y) \Leftrightarrow (\forall p\ p(x)=p(y))$
      or "2 functions are equal iff they have the same value for all args":
      $\forall f,g\ \ (f=g) \Leftrightarrow (\forall x\ \ f(x)=g(x))$
    - Problem: inference procedures not well understood.
- **λ-expressions**
  - "Macros" to construct complex predicates and functions
    - e.g.Definition: $\lambda x,y\ \ King(x) \wedge Brother(x,y)$
      Usage: $(\lambda x,y\ King(x) \wedge Brother(x,y))\ (Richard, John)$

©1996-2006 Michel Pasquier. All rights reserved

Using First-Order Logic

- **Knowledge domain**
  - A part of the world we want to express knowledge about
- **Example of the kinship domain**
  - Objects: people e.g., Elizabeth, Charles, William, etc.
  - Properties: gender i.e., male, female
    Unary predicates: Male() and Female()
  - Relations: kinship e.g., motherhood, brotherhood, etc.
    Binary predicates: Parent(), Sibling(), Brother(), Child(), etc.
    Functions: MotherOf(), FatherOf()
  - > Express facts e.g., Charles is a male
    and rules e.g., the mother of a parent is a grandmother

©1996-2006 Michel Pasquier. All rights reserved

Sample Functions and Predicates

- **Functions**

  $\forall$ x,y FatherOf(x)=y $\Leftrightarrow$ Parent(y,x) $\wedge$ Male(y)

  $\forall$ x,y MotherOf(x)=y $\Leftrightarrow$ Parent(y,x) $\wedge$ Female(y)

- **Predicates**

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

  $\forall$ x,y Grandparent(x,y) $\Leftrightarrow$ $\exists$ z, Parent(x,z) $\wedge$ Parent(z,y)

  $\forall$ x,y Sibling(x,y) $\Leftrightarrow$ $\neg$x=y $\wedge$ $\exists$ z, Parent(z,x) $\wedge$ Parent(z,y)

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

- **Potential problems**
  - Self-definition (causes infinite recursion)
    - e.g.: $\forall$ x,y Child(x,y) $\Leftrightarrow$ Parent(y,x) following the above

©1996-2006 Michel Pasquier. All rights reserved

TELLing and ASKing

- **TELLing the KB**
  - Assertion: add a sentence to the knowledge base
    - e.g.
      TELL(KB, $\forall x,y$ MotherOf(x)=y $\Leftrightarrow$ Parent(y,x) $\wedge$ Female(y))
      and so on, then
      TELL(KB, Female(Elizabeth) $\wedge$ Parent(Elizabeth,Charles) $\wedge$ Parent(Charles,William))
- **ASKing the KB**
  - Query: retrieve/infer a sentence from the knowledge base
  - Yes/no answer
    - e.g. ASK(KB, Grandparent(Elizabeth, William))
  - Binding list, or <u>substitution</u>
    - e.g. ASK(KB, $\exists x$ Child(William, x)) yields {x / Charles}

©1996-2006 Michel Pasquier. All rights reserved

A Logical Agent for the Wumpus World

- **TELLing the KB**
  - Interface: percepts + actions
  - Percept sentences
    - e.g. Percept([Stench, Breeze, Glitter, None, None], t)
  - Action sentences
    - e.g. Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb
- **ASKing the KB**
  - Queries
    - e.g. ∃a Action(a, t+1)
      returning a binding list, such as {a / Grab}

©1996-2006 Michel Pasquier. All rights reserved

Summary

- **First-order logic …**
  - Is a general-purpose knowledge representation language.
  - Is based on the ontological commitment that the world is composed of objects, with properties and relations.
- **The syntax of first-order logic …**
  - Constant symbols name objects.
  - Predicate symbols name relations.
  - Complex terms name objects using function symbols.
  - Atomic sentences consist of predicates applied to terms.
  - Complex sentences use connectives.
  - Quantified sentences allow to express general rules.

©1996-2006 Michel Pasquier. All rights reserved

Summary

- **Knowledge-based agents can …**
  - Be designed using first-order logic.
  - Reason using first-order logic.
- **Knowledge-based agents need to …**
  - React to what they perceive.
  - Abstract descriptions of states from percepts.
  - Maintain an internal model of the relevant aspects of the world not directly available from percepts.
  - Express and use information about the desirability of their actions.
  - Use goals in conjunction with knowledge to make plans.

©1996-2006 Michel Pasquier. All rights reserved

References

- Bell, J. L. and Machover, M. (1977). *A Course in Mathematical Logic*. Elsevier/North-Holland, Amsterdam, London, New York.
- Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press, New York.
- Kowalski, R. (1979b). *Logic for Problem Solving*. Elsevier/North-Holland, Amsterdam, London, New York.
- Quine, W. V. (1982). *Methods of Logic*. Harvard University Press, Cambridge, Massachusetts, fourth edition.
- Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81-132.

©1996-2006 Michel Pasquier. All rights reserved

 ©1996-2006 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - Frames and Semantic / Conceptual Networks.
  - Managing Retractions, Assumptions, and Explanations.

©1996-2006 Michel Pasquier. All rights reserved

9 – INFERENCE IN FIRST-ORDER LOGIC

“In which we define inference mechanisms that can efficiently answer questions posed in first-order logic.”

©1996-2006 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - – Inference Rules. – Generalised Modus Ponens.
  - – Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - – Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - – Frames and Semantic / Conceptual Networks.
  - – Managing Retractions, Assumptions, and Explanations.

©1996-2006 Michel Pasquier. All rights reserved

Inferences Rules for FOL

- **Inference rules from Propositional Logic**
  - *Modus Ponens*
    - $$\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$$
  - Double-Negation-Elimination
    - $$\frac{\neg\neg\alpha}{\alpha}$$
  - And-Elimination
    - $$\frac{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}{\alpha_i}$$
  - And-Introduction
    - $$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}$$
  - Or-Introduction
    - $$\frac{\alpha_i}{\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n}$$
  - Resolution
    - $$\frac{\alpha \lor \beta,\ \neg\beta \lor \gamma}{\alpha \lor \gamma}$$

©1996-2006 Michel Pasquier. All rights reserved

Inferences Rules with Quantifiers

- **Substitutions**
  - SUBST(θ , α): binding list θ applied to a sentence α
    - e.g.: SUBST({x / John, y / Richard}, Brother(x, y)) = Brother(John, Richard)

- **Inference rules**
  - Universal Elimination
    - $$\frac{\forall\, x\ \ \alpha}{\text{SUBST}(\ \{x/g\},\ \alpha)}$$

  $\forall$ x Dog(x) $\Rightarrow$ Friendly(x)

  |– Dog(Snoopy) $\Rightarrow$ Friendly(Snoopy)

  - Existential Introduction
    - $$\frac{\alpha}{\exists\ x\ \text{SUBST}(\ \{g/v\},\ \alpha)}$$

  - Existential Elimination
    - $$\frac{\exists\ \ x\ \ \alpha}{\text{SUBST}(\ \{x/\ K\},\ \alpha)}$$

  (Skolemization)

  $\exists$x Dog (x) Λ Owns(John, x)

  |– Dog (Lassie), Owns(John,Lassie)

©1996-2006 Michel Pasquier. All rights reserved

An Example of Logical Proof

- **Proof procedure**
  - Analysis of the problem description (Natural Language)
  - Translation from NL to first-order logic
  - Application of inference rules (proof)
- **Problem statement**
  - *"It is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold by Col. West, who is American."*

©1996-2006 Michel Pasquier. All rights reserved

end

- I Artificial Intelligence
- II Problem Solving
- **III Knowledge and Reasoning**
- IV Acting Logically
- V Uncertain Knowledge and Reasoning
- VI Learning
- VII Communicating, Perceiving and Acting
- VIII Conclusions

©1996-2006 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - Frames and Semantic / Conceptual Networks.
  - Managing Retractions, Assumptions, and Explanations.

©1996-2006 Michel Pasquier. All rights reserved

9 – INFERENCE IN FIRST-ORDER LOGIC

"In which we define inference mechanisms that can efficiently answer questions posed in first-order logic."

©1996-2006 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - Frames and Semantic / Conceptual Networks.
  - Managing Retractions, Assumptions, and Explanations.

©1996-2006 Michel Pasquier. All rights reserved

Inferences Rules for FOL

- **Inference rules from Propositional Logic**
  - *Modus Ponens*
    - $$\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$$
  - Double-Negation-Elimination
    - $$\frac{\neg\neg\alpha}{\alpha}$$
  - And-Elimination
    - $$\frac{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}{\alpha_i}$$
  - And-Introduction
    - $$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}$$
  - Or-Introduction
    - $$\frac{\alpha_i}{\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n}$$
  - Resolution
    - $$\frac{\alpha \lor \beta,\ \neg\beta \lor \gamma}{\alpha \lor \gamma}$$

©1996-2006 Michel Pasquier. All rights reserved

Inferences Rules with Quantifiers

- **Substitutions**
  - SUBST(θ , α): binding list θ applied to a sentence α
    - e.g.: SUBST({x / John, y / Richard}, Brother(x, y)) = Brother(John, Richard)

- **Inference rules**
  - Universal Elimination
    - $$\frac{\forall\, x \;\; \alpha}{\text{SUBST}(\{x/g\}, \alpha)}$$

$\forall$ x Dog(x) $\Rightarrow$ Friendly(x)

|– Dog(Snoopy) $\Rightarrow$ Friendly(Snoopy)

  - Existential Introduction
    - $$\frac{\alpha}{\exists\, x\ \text{SUBST}(\{g/v\}, \alpha)}$$

  - Existential Elimination
    - $$\frac{\exists\;\; x \;\; \alpha}{\text{SUBST}(\{x/ K\}, \alpha)}$$

(Skolemization)

$\exists$x Dog (x) Λ Owns(John, x)

|– Dog (Lassie), Owns(John,Lassie)

©1996-2006 Michel Pasquier. All rights reserved

An Example of Logical Proof

- **Proof procedure**
  - Analysis of the problem description (Natural Language)
  - Translation from NL to first-order logic
  - Application of inference rules (proof)
- **Problem statement**
  - *"It is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold by Col. West, who is American."*

©1996-2006 Michel Pasquier. All rights reserved

"Webscape, a competitor of Macrosoft, developed some nice software, all of which was stolen by Bill, who is a CEO. It is immoral for a CEO to steal business from rival companies. A competitor of Macrosoft is a rival. Software is business."

immoral ← criminal
CEO ← American
steal ← sell
business ← weapon
rival ← hostile
company ← nation
Webscape ← Nono
competitor ← enemy
Macrosoft ← America
software ← missile
develop ← own
Bill ← West

$Competitor(Webscape, Macrosoft)$

$\exists x\ Software(x) \land Developed(Webscape, x)$

$\forall x\ Software(x) \land Developed(Webscape, x) \Rightarrow Steal(Bill, x, Webscape)$

$CEO(Bill)$

$\forall x,y,z\ CEO(x) \land Business(y) \land Company(z) \land Rival(z) \land Steal(x,y,z) \Rightarrow Immoral(x)$

$\forall x\ Competitor(x, Macrosoft) \Rightarrow Rival(x)$

$\forall x\ Software(x) \Rightarrow Business(x)$

$Company(Webscape)$

$Company(Macrosoft)$

©1996-2006 Michel Pasquier. All rights reserved

Horn Clauses (Prolog style)
Competitor (Webscape, Macrosoft).
(2a) Software(Mozilla).   // Skolem constant
(2b) Developed(Webscape, Mozilla).
Steal(Bill, x, Webscape) :– Software(x), Developed(Webscape, x).
CEO(bill).
Immoral(x) :– CEO(x), Business(y), Company(z), Rival(z), Steal(x,y,z).
Rival(x) :– Competitor(x, Macrosoft).
Business(x) :– Software(x).
Company(Webscape).
Company(Macrosoft).

Proof using Modus Ponens
→ Bill is immoral

From 2-a, 7, and Modus Ponens, w/ { x/Mozilla }:
(10) Business(Mozilla)

From 1, 6, and Modus Ponens, w/ { x/Webscape}:
(11) Rival(Webscape)

From 2-a, 2-b, 3, and Modus Ponens, w/ { x/Mozilla }::
(12) Steal(Bill, Mozilla, Webscape)

From 4, 10, 8, 11, 12, 5, and Modus Ponens, w/ { x/Bill, y/Mozilla, z/Webscape}:
(13) Immoral(Bill)

©1996-2006 Michel Pasquier. All rights reserved

Translation in First-Order-Logic

- "It is a crime for an American to sell weapons to hostile nations …"
  - **(1)** $\forall x,y,z\ American(x) \land Weapon(y) \land Nation(z) \land Hostile(z) \land Sells(x,z,y) \Rightarrow Criminal(x)$
- "The country Nono […] has some missiles, …"
  - **(2)** $\exists x\ Owns(Nono, x) \land Missile(x)$
- "… all of its missiles were sold by Col. West, …"
  - **(3)** $\forall x\ Owns(Nono,x) \land Missile(x) \Rightarrow Sells(West,Nono,x)$
- A missile is a weapon.
  - **(4)** $\forall x\ Missile(x) \Rightarrow Weapon(x)$
- An enemy of America is hostile.
  - **(5)** $\forall x\ Enemy(x,America) \Rightarrow Hostile(x)$
- "… West, who is American."
  - **(6)** $American(West)$
- "The country Nono …"
  - **(7)** $Nation(Nono)$
- "Nono, an enemy of America …"
  - **(8)** $Enemy(Nono,America)$
  - **(9)** $Nation(America)$

©1996-2006 Michel Pasquier. All rights reserved

Proof

Knowledge Base

(1) $\forall x,y,z$ American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

(2) $\exists x$ Owns(Nono, x) $\wedge$ Missile(x)

(3) $\forall x$ Owns(Nono,x) $\wedge$ Missile(x) $\Rightarrow$ Sells(West,Nono,x)

(4) $\forall x$ Missile(x) $\Rightarrow$ Weapon(x)

(5) $\forall x$ Enemy(x,America) $\Rightarrow$ Hostile(x)

(6) American(West)

(7) Nation(Nono)

(8) Enemy(Nono,America)

(9) Nation(America)

Inferences

From (2) and Existential-Elimination:

(10) Owns(Nono, M1) $\wedge$ Missile(M1)

From (10) and And-Elimination:

(11) Owns(Nono, M1)

(12) Missile(M1)

From (4) and Universal-Elimination:

(13) Missile(M1) $\Rightarrow$ Weapon(M1)

From (12,13) and Modus Ponens:

(14) Weapon(M1)

©1996-2006 Michel Pasquier. All rights reserved

Proof (2)

Knowledge Base

(1) $\forall$x,y,z American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

…

(3) $\forall$x Owns(Nono,x) $\wedge$ Missile(x) $\Rightarrow$ Sells(West,Nono,x)

…

(6) American(West)

…

(10) Owns(Nono, M1) $\wedge$ Missile(M1)

…

(14) Weapon(M1)

Inferences

From (3) and Universal-Elimination:

(15) Owns(Nono, M1) $\wedge$ Missile(M1) $\Rightarrow$ Sells(West,Nono,M1)

From (15,10) and Modus Ponens:

(16) Sells(West,Nono,M1)

From (1) and Universal-Elimination (three times):

(17) American(West) $\wedge$ Weapon(M1) $\wedge$ Nation (Nono) $\wedge$ Hostile(Nono) $\wedge$ Sells(West,Nono,M1) $\Rightarrow$ Criminal(West)

©1996-2006 Michel Pasquier. All rights reserved

Proof (3)

Knowledge Base

...

(5) $\forall x$ Enemy(x,America) $\Rightarrow$ Hostile(x)

(6) American(West)

(7) Nation(Nono)

(8) Enemy(Nono,America)

...

(14) Weapon(M1)

(16) Sells(West,Nono,M1)

(17) American(West) $\wedge$ Weapon(M1) $\wedge$ Nation (Nono) $\wedge$ Hostile(Nono) $\wedge$ Sells(West,Nono,M1) $\Rightarrow$ Criminal(West)

Inferences

From (5) and Universal-Elimination:

(18) Enemy(Nono,America) $\Rightarrow$ Hostile(Nono)

From (8,18) and Modus Ponens:

(19) Hostile(Nono)

From (6,7,14,16,19) and And-Intro.:

(20) American(West) $\wedge$ Weapon(M1) $\wedge$ Nation (Nono) $\wedge$ Hostile(Nono) $\wedge$ Sells(West,Nono,M1)

From (17,20) and Modus Ponens:

(21) **Criminal(West)**

©1996-2006 Michel Pasquier. All rights reserved

Proof as a Search Problem

- **Proof procedure**
  - Sequence of inference rules applied to the KB
- **Search problem formulation**
  - Initial state: KB (sentences 1 to 9)
  - Operators: applicable inference rules
  - Goal state: KB containing Criminal(West)
- **Characteristics**
  - Solution depth: 14
  - Branching factor increases as the KB grows, very large for some operators (e.g. Universal Elimination)
  - Common inference patterns (using U.E., A.I., M.P.)

©1996-2006 Michel Pasquier. All rights reserved

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - Frames and Semantic / Conceptual Networks.
  - Managing Retractions, Assumptions, and Explanations.

©1996-2006 Michel Pasquier. All rights reserved

Generalised Modus Ponens

- **Inference pattern**
  - Universal-Elimination + And-Introduction + Modus Ponens
    - e.g.: $\forall x$ Missile(x) $\land$ Owns(Nono,x) $\Rightarrow$ Sells(West,Nono,x)
      Missile(M1)
      Owns(Nono,M1) $\vdash$ Sells(West,Nono,M1)

- **Inference rule**
  - *Generalised Modus Ponens*
    - $$\frac{\chi_1, \chi_2, \ldots, \chi_N, (\alpha_1 \land \alpha_2 \land \ldots \land \alpha_N \Rightarrow \beta)}{\text{SUBST}(\theta, \beta)}$$

      where $\forall i$ SUBST(θ, χ_i) = SUBST(θ, α_i)

©1996-2006 Michel Pasquier. All rights reserved

Example of GMP Application

- **Knowledge base (extract)**
  - $\forall x$ Missile(x) Λ Owns(Nono,x) $\Rightarrow$ Sells(West,Nono,x)
  - $\forall y$ Owns(y,M1)
  - Missile(M1)
- **Generalized Modus Ponens**
  - Matching
    - $\chi_1 \leftarrow$ Missile(M1)
    - $\chi_2 \leftarrow$ Owns(y,M1)
    - $\theta \leftarrow$ {x/M1, y/Nono)
    - $\alpha_1 \leftarrow$ Missile(x)
    - $\alpha_2 \leftarrow$ Owns(Nono,x)
    - $\beta \leftarrow$ Sells(West,Nono,x)
  - Inference rule
    - $\dfrac{\chi_1, \chi_2, \ (\alpha_1 \Lambda \alpha_2 \Rightarrow \beta)}{\mathrm{SUBST}(\theta, \beta)}$ $\leftarrow$ Sells(West,Nono,M1)

©1996-2006 Michel Pasquier. All rights reserved

Using the GMP

- **Characteristics**
  - Combine several inferences into one
  - Use helpful substitutions (rather than random U.E.)
  - Make use of pre-compiled rules in…
- **Canonical form**
  - Matches the premises of the GMP rule
  - Horn sentences (Horn normal forms / clause forms)
    - i.e. $\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \Rightarrow \beta$
  - Sentences converted when entered in the KB
    - e.g. $\exists x$ Missile(x) $\land$ Owns(Nono,x) becomes
      Missile(M66) and Owns(Nono,M66)

©1996-2006 Michel Pasquier. All rights reserved

Unification

- **The UNIFY routine**
  - Find a substitution that make 2 atomic sentences alike i.e.

    $$\textbf{UNIFY}(\alpha,\beta) = \underset{\text{unifier}}{\theta} \quad \text{where} \quad \text{SUBST}(\theta,\alpha) = \text{SUBST}(\theta,\beta)$$

- **Example**
  - Sample rule in canonical form:
    - Knows(John,x) $\Rightarrow$ Hates(John,x)
  - Query: “who does John hate?”
    - ?p, Hates(John,p)
    - Find all the sentences in the KB that unify with Knows(John,x), then apply the unifier to Hates(John,x).

©1996-2006 Michel Pasquier. All rights reserved

Variable Substitution

- **Renaming**
  - Sentence identical to another, except for variable names
    - e.g. Hates(x,Elizabeth) and Hates(y,Elizabeth)
- **Composition of substitutions**
  - *Substitution with composed unifier identical to the sequence of substitutions with each unifier*

    i.e.

    $$\text{Subst}(\text{Compose}(\theta_1,\theta_2),\alpha) = \text{Subst}(\theta_2\ ,\text{Subst}(\theta_1,\alpha))$$

    - e.g. α = Knows(x,y), θ_1 = {x/John}, θ_2 = {y/ Elizabeth}
      Subst(θ_2,Subst(θ_1 ,α)) = Subst(θ_2,Knows(John,y)) =
      Subst({x/John,y/Elizabeth}, Knows(x,y)) = Knows(John,Elizabeth)

©1996-2006 Michel Pasquier. All rights reserved

Standardising Sentences

- **Example**
  - Knowledge base:
    - Knows(John,Jane)
    - Knows(y,Leonid)
    - Knows(z,Mother(z))
    - Knows(x,Elizabeth)
  - Unifying with Knows(John,x):
    - UNIFY(Knows(John,x), Knows(John,Jane)) = {x/Jane}
    - UNIFY(Knows(John,x), Knows(y,Leonid)) = {x/Leonid, y/John}
    - UNIFY(Knows(John,x), Knows(z,Mother(z))) = {z/John, x/Mother(John)}
    - UNIFY(Knows(John,x), Knows(x,Elizabeth)) = {} ?
- **Standardise sentences apart**
  - Renaming variables to avoid clashes, e.g. Knows(z,Elizabeth)

©1996-2006 Michel Pasquier. All rights reserved

Most General Unifier

- **Example**
  - Unifying yields an infinite number of substitutions
    - UNIFY(Knows(John,x), Knows(z,Elizabeth)) = {x/Elizabeth, z/John}
      or {x/Elizabeth, z/John, w/Richard},
      or {x/Elizabeth, y/Elizabeth, z/John},
      or …
- **Most General Unifier (MGU)**
  - *Unifier that makes the least commitments about the bindings of the variables*
  - UNIFY always returns the MGU

©1996-2006 Michel Pasquier. All rights reserved

Sample Proof Revisited

Knowledge Base

(1) $\forall x,y,z$ American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

(2) $\exists x$ Owns(Nono, x) $\wedge$ Missile(x)

(3) $\forall x$ Owns(Nono,x) $\wedge$ Missile(x) $\Rightarrow$ Sells(West,Nono,x)

(4) $\forall x$ Missile(x) $\Rightarrow$ Weapon(x)

(5) $\forall x$ Enemy(x,America) $\Rightarrow$ Hostile(x)

(6) American(West)

(7) Nation(Nono)

(8) Enemy(Nono,America)

(9) Nation(America)

KB in Horn Normal Form

(1) American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

(2a) Owns(Nono, M1)

(2b) Missile(M1)

(3) Owns(Nono,x) $\wedge$ Missile(x) $\Rightarrow$ Sells(West,Nono,x)

(4) Missile(x) $\Rightarrow$ Weapon(x)

(5) Enemy(x,America) $\Rightarrow$ Hostile(x)

(6) American(West)

(7) Nation(Nono)

(8) Enemy(Nono,America)

(9) Nation(America)

©1996-2006 Michel Pasquier. All rights reserved

Sample Proof (2)

Knowledge Base (HNF)

(1) American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

(2a) Owns(Nono, M1)

(2b) Missile(M1)

(3) Owns(Nono,x) $\wedge$ Missile(x) $\Rightarrow$ Sells(West,Nono,x)

(4) Missile(x) $\Rightarrow$ Weapon(x)

(5) Enemy(x,America) $\Rightarrow$ Hostile(x)

(6) American(West)

(7) Nation(Nono)

(8) Enemy(Nono,America)

(9) Nation(America)

Inferences

From (2b, 4) and Modus Ponens:

(10) Weapon(M1)

From (8, 5) and Modus Ponens:

(11) Hostile(Nono)

From (2a, 2b, 3) and Modus Ponens:

(12) Sells(West,Nono,M1)

From (6, 10, 7, 11, 12, 1) and Modus Ponens:

(13) **Criminal(West)**

©1996-2006 Michel Pasquier. All rights reserved

end

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - – Inference Rules. – Generalised Modus Ponens.
  - – Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - – Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - – Frames and Semantic / Conceptual Networks.
  - – Managing Retractions, Assumptions, and Explanations.

©1996-2006 Michel Pasquier. All rights reserved

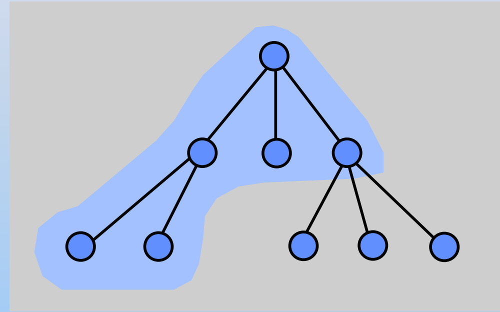
Forward and Backward Chaining

- **Reasoning**
  - Knowledge representation language (First-Order Logic)
  - Efficient inference rule (Generalised Modus Ponens)
  - > *Generate the proof*
- **Using the GMP**
  - <u>Forward chaining</u> (data-driven): KB, α |– ?
    - Start with the KB and generate new sentences
      e.g. to derive the consequences of newly added facts.
  - <u>Backward chaining</u> (goal-driven): KB |– α?
    - Start with a sentence not in the KB and attempt to establish
      its premises, e.g. to prove some new fact.

©1996-2006 Michel Pasquier. All rights reserved

Forward Chaining

- **Idea: inferring consequences**
  - TELLing a new sentence α, KB, α |– ?



- **Pseudo-algorithm**
  - If α already in the KB, do nothing
  - Find all implications that have α as a premise, i.e.
    $\alpha \wedge \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta$ then
  - If all other premises α_i are known under some MGU θ, infer the conclusion β under θ
  - If some premises α_i can be matched several ways, then infer each corresponding conclusion

©1996-2006 Michel Pasquier. All rights reserved

Variable Substitution

- **Renaming**
  - Sentence identical to another, except for variable names
    - e.g. Hates(x,Elizabeth) and Hates(y,Elizabeth)
- **Composition of substitutions**
  - *Substitution with composed unifier identical to the sequence of substitutions with each unifier*

    i.e.

    $$\text{Subst}(\text{Compose}(\theta_1,\theta_2),\alpha) = \text{Subst}(\theta_2, \text{Subst}(\theta_1,\alpha))$$

    - e.g. α = Knows(x,y), θ_1 = {x/John}, θ_2 = {y/ Elizabeth}
      Subst(θ_2,Subst(θ_1 ,α)) = Subst(θ_2,Knows(John,y)) =
      Subst({x/John,y/Elizabeth}, Knows(x,y)) = Knows(John,Elizabeth)

©1996-2006 Michel Pasquier. All rights reserved

Example of Forward Chaining

Knowledge Base (HNF)

(1) American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

(2) Owns(Nono,x) $\wedge$ Missile(x) $\Rightarrow$ Sells(West,Nono,x)

(3) Missile(x) $\Rightarrow$ Weapon(x)

(4) Enemy(x,America) $\Rightarrow$ Hostile(x)

Adding Atomic Sentences

Forward-Chain(KB, American(West)):
(5) American(West)
Unifies with a premise of (1), others not known: no new inference.

Forward-Chain(KB, Nation(Nono)):
(6) Nation(Nono) id.

Forward-Chain(KB, Enemy(Nono,America)):
(7) Enemy(Nono,America)
Unifies with (4), with unifier {x/Nono}; call …

©1996-2006 Michel Pasquier. All rights reserved

Knowledge Base (HNF)

(1) American(x) Λ Weapon(y) Λ Nation (z) Λ Hostile(z) Λ Sells(x,z,y) ⇒ Criminal(x)

(2) Owns(Nono,x) Λ Missile(x) ⇒ Sells(West,Nono,x)

(3) Missile(x) ⇒ Weapon(x)

(4) Enemy(x,America) ⇒ Hostile(x)

(5) American(West)

(6) Nation(Nono)

(7) Enemy(Nono,America)

Inferences

Forward-Chain(KB, Hostile(Nono)):
(8) Hostile(Nono)
Unifies with (1), no new inference.

Forward-Chain(KB, Owns(Nono,M1)):
(9) Owns(Nono,M1)
Unifies with (2), no new inference.

Forward-Chain(KB, Missile(M1)):
(10) Missile(M1)
Unifies with (2), with unifier {x/M1}.
Other premise known; call
Forward-Chain(KB, Sells(West, Nono,M1)):

©1996-2006 Michel Pasquier. All rights reserved

Knowledge Base (HNF)

(1) American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

(2) Owns(Nono,x) $\wedge$ Missile(x) $\Rightarrow$ Sells(West,Nono,x)

(3) Missile(x) $\Rightarrow$ Weapon(x)

(4) Enemy(x,America) $\Rightarrow$ Hostile(x)

(5) American(West)

(6) Nation(Nono)

(7) Enemy(Nono,America)

(8) Hostile(Nono)

(9) Owns(Nono,M1)

(10) Missile(M1)

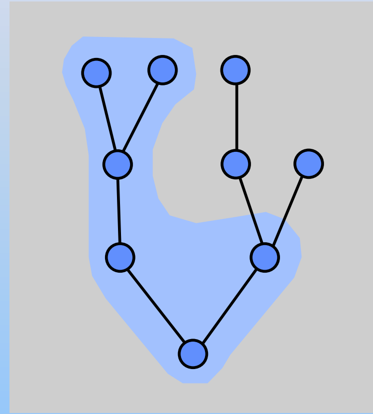
Inferences

(11) Sells(West,Nono,M1)
Unifies with (1), no new inference.

Back to **(10)** Missile(M1):
Unifies with (3), w/ unifier {x/M1}; call Forward-Chain(KB, Weapon(M1)):
(12) Weapon(M1)
Unifies with (1), all other premises known, with {x/West, y/M1, z/Nono};
Call
Forward-Chain(KB, Criminal(West)):
(13) **Criminal(West)**

©1996-2006 Michel Pasquier. All rights reserved

Backward Chaining



- **Idea: checking for causes**
  – ASKing a query β, KB |– β?
- **Pseudo-algorithm**
  – If β already in the KB, proof immediate
  – Find all implications that have β as a conclusion, i.e.
  $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n \Rightarrow \beta$ then
  – Try and establish all the (qlist) premises [α_i] then infer β

function Backward-Chain (KB, β) **returns** substitutions
  return Back-Chain-List(KB, [β], { })

©1996-2006 Michel Pasquier. All rights reserved

Example of Backward Chaining

Knowledge Base (HNF)

(1) American(x) Λ Weapon(y) Λ Nation (z) Λ Hostile(z) Λ Sells(x,z,y) ⇒ Criminal(x)

(2a) Owns(Nono, M1)

(2b) Missile(M1)

(3) Owns(Nono,u) Λ Missile(u) ⇒ Sells(West,Nono,u)

(4) Missile(v) ⇒ Weapon(v)

(5) Enemy(c,America) ⇒ Hostile(c)

(6) American(West)

(7) Nation(Nono)

(8) Enemy(Nono,America)

(9) Nation(America)

Establishing Premises

Backward-Chain(KB,Criminal(West)):
Call
B-Chain-List(KB, [Criminal(West)], {})

Unifies with conclusion of (1);
answers = {x/West}; call
B-Chain-List(KB, [American(West), Weapon(y), Nation(z), Hostile(z), Sells(West,z,y)], {x/West}):

American(West) as **(6)**;
Weapon(y) unifies with conclusion of (4) …

©1996-2006 Michel Pasquier. All rights reserved

Knowledge Base (HNF)

(1) American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

(2a) Owns(Nono, M1)

(2b) Missile(M1)

(3) Owns(Nono,u) $\wedge$ Missile(u) $\Rightarrow$ Sells(West,Nono,u)

(4) Missile(v) $\Rightarrow$ Weapon(v)

(5) Enemy(c,America) $\Rightarrow$ Hostile(c)

(6) American(West)

(7) Nation(Nono)

(8) Enemy(Nono,America)

(9) Nation(America)

Establishing Premises

answers = {x/West, y/v}; call
B-Chain-List(KB, [Missile(v)],
{x/West, y/v}):

Missile(v) unifies with **(2b)**;
answers = {x/West, y/M1};
Weapon(M1) established; back, call

B-Chain-List(KB, [Nation(z)],
{x/West, y/M1}):

Nation(z) unifies with **(7)**;
answers = {x/West, y/M1, z/Nono};
Hostile(Nono) unifies with conclusion of **(5)**; call …

©1996-2006 Michel Pasquier. All rights reserved

Knowledge Base (HNF)

(1) American(x) $\wedge$ Weapon(y) $\wedge$ Nation (z) $\wedge$ Hostile(z) $\wedge$ Sells(x,z,y) $\Rightarrow$ Criminal(x)

(2a) Owns(Nono, M1)

(2b) Missile(M1)

(3) Owns(Nono,u) $\wedge$ Missile(u) $\Rightarrow$ Sells(West,Nono,u)

(4) Missile(v) $\Rightarrow$ Weapon(v)

(5) Enemy(c,America) $\Rightarrow$ Hostile(c)

(6) American(West)

(7) Nation(Nono)

(8) Enemy(Nono,America)

(9) Nation(America)

Establishing Premises

B-Chain-List(KB, [Enemy(Nono, America)], {x/West, y/M1, z/Nono}):

Enemy(Nono,America) as **(8)**;
Hostile(Nono) established; back, call

B-Chain-List(KB, [Sells(West,Nono, M1), {x/West, y/M1, z/Nono}):

Sells(West,Nono,M1) unifies with **(3)**; call
B-Chain-List(KB,[Owns(Nono,M1)] …)

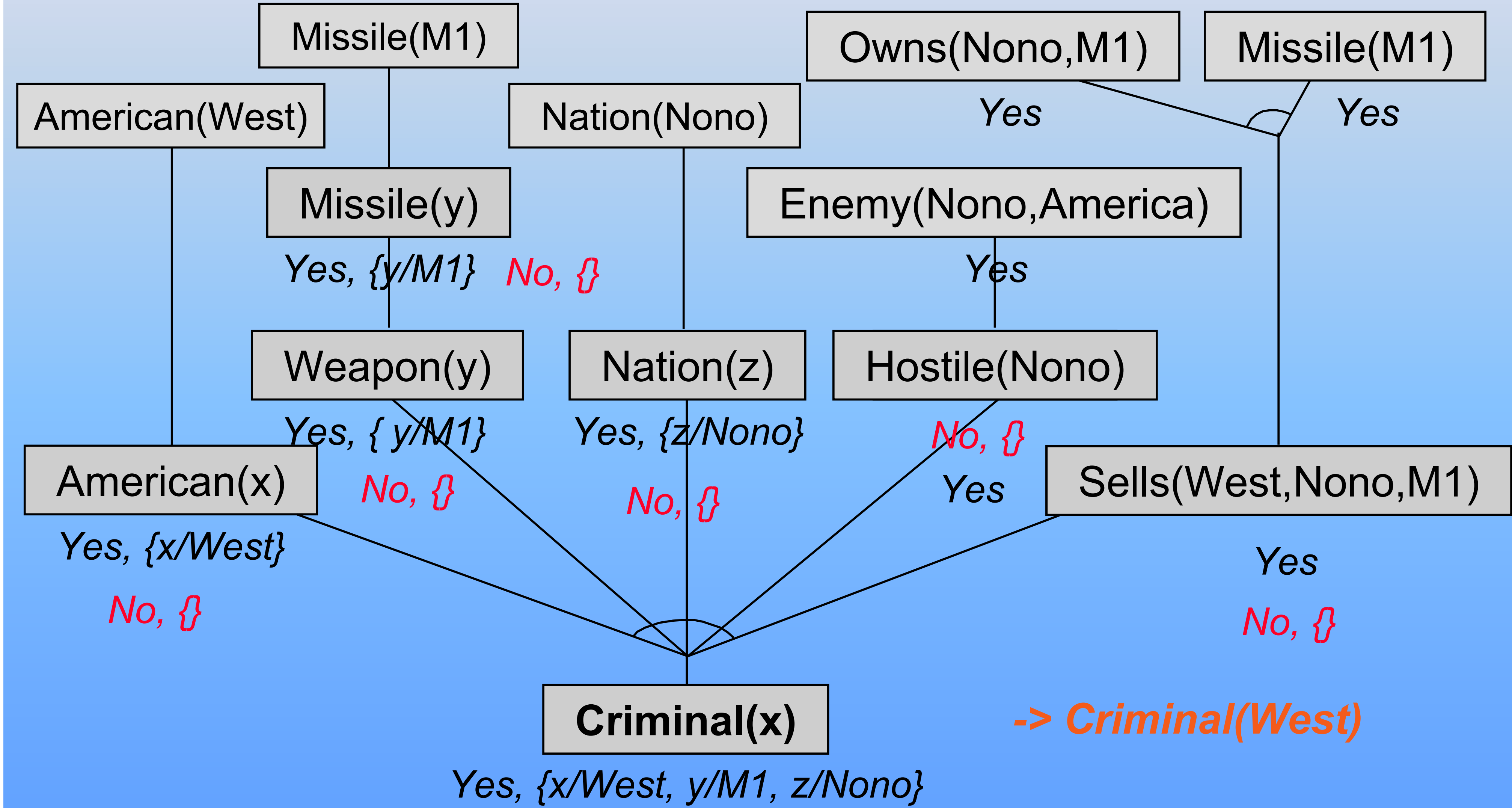
Owns(Nono,M1) as **(2a)**;

B-Chain-List(KB,[Missile(M1)], …)

Missile(M1) as **(2b)**. **end**

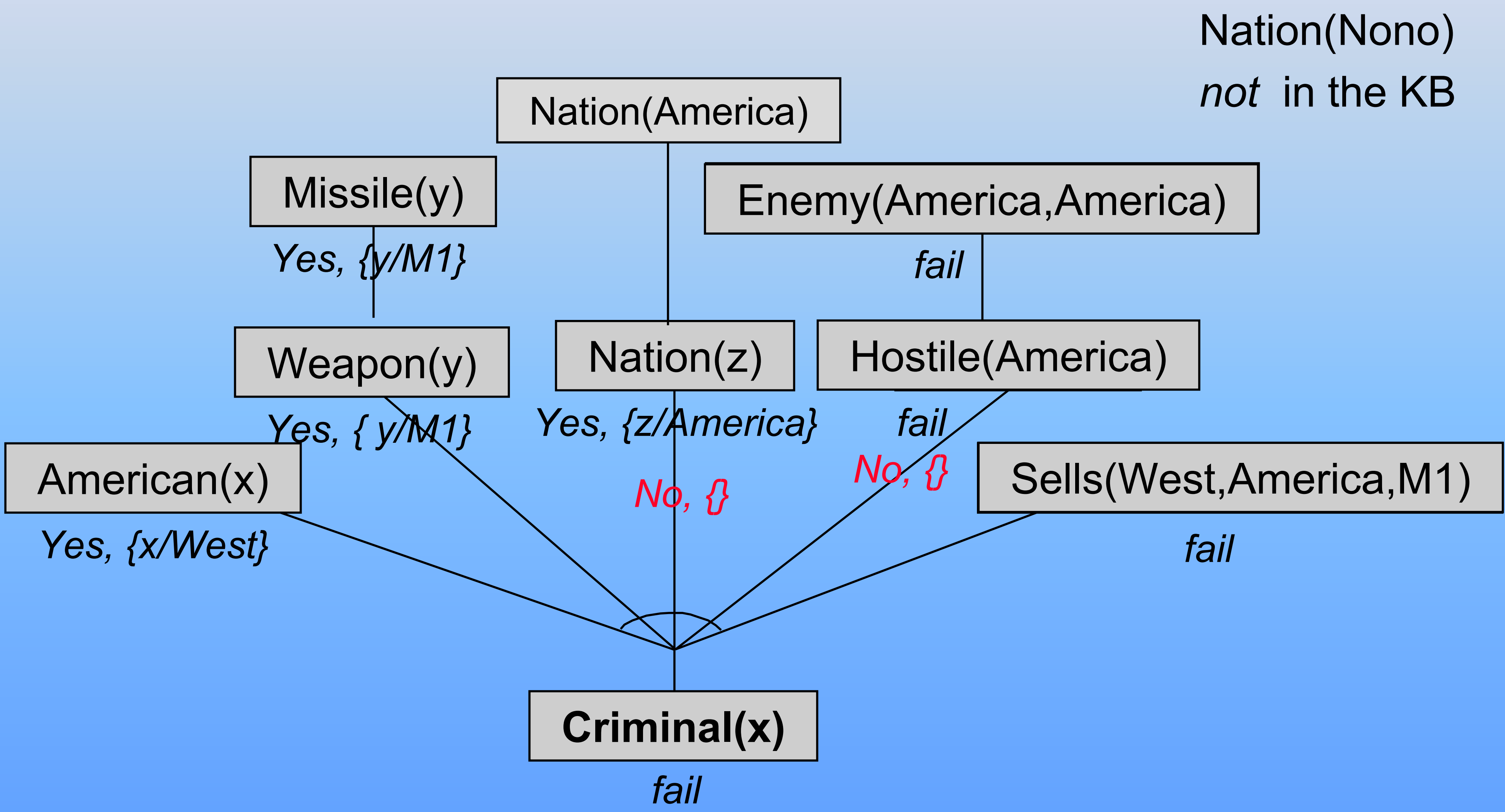
©1996-2006 Michel Pasquier. All rights reserved

Chaining as a Proof Tree
(with Success)



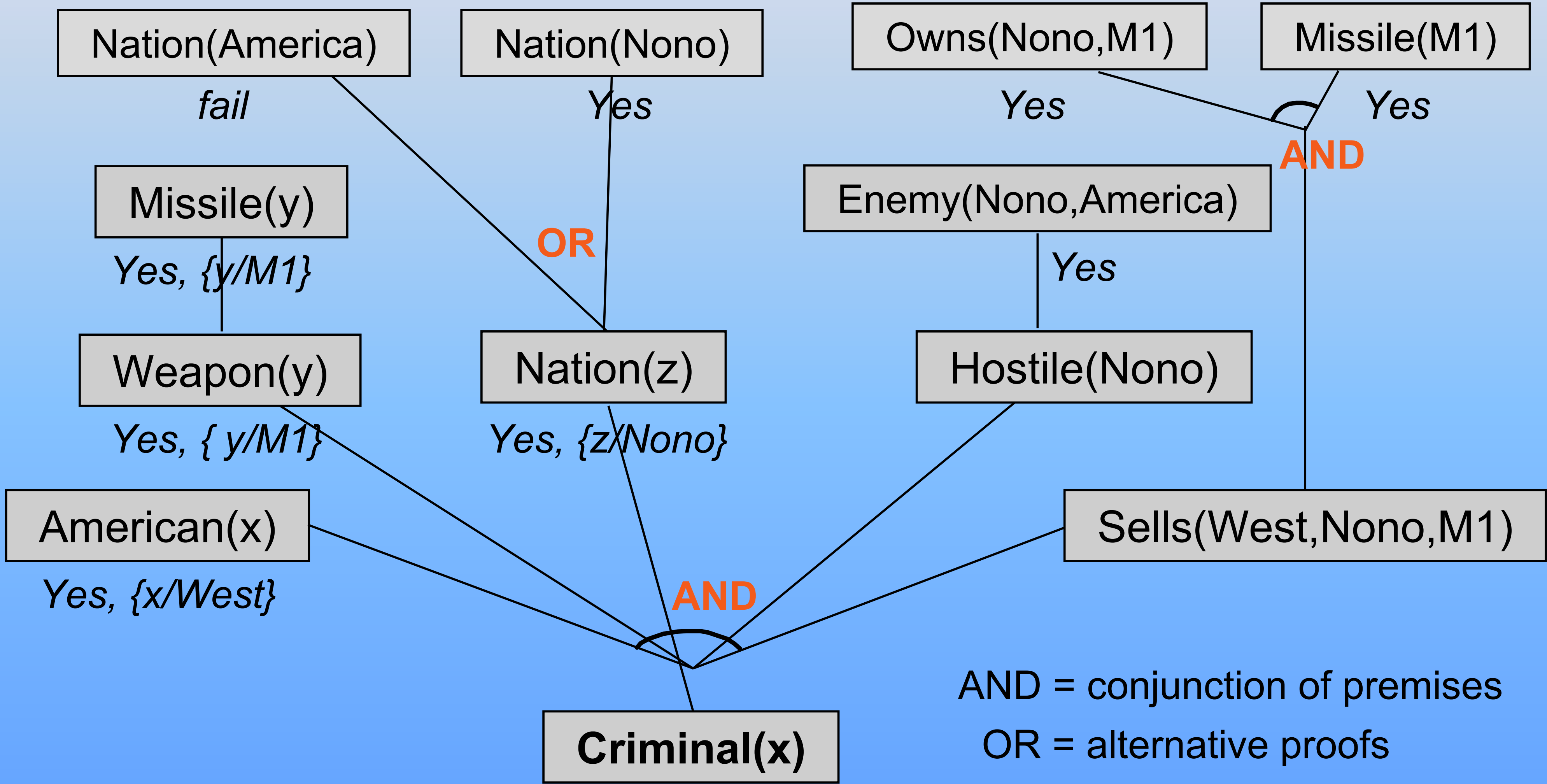
©1996-2006 Michel Pasquier. All rights reserved

Chaining as a Proof Tree
(with Failure)



©1996-2006 Michel Pasquier. All rights reserved

And-Or Proof Tree
(with Backtracking)



AND = conjunction of premises
OR = alternative proofs

©1996-2006 Michel Pasquier. All rights reserved

Completeness

- **GMP is not complete**
  - There exist entailed sentences that GMP cannot prove
    - Some sentences cannot be converted to Horn sentences e.g.

KB:	$\forall x\ P(x) \Rightarrow Q(x)$	$\forall x\ Q(x) \Rightarrow S(x)$	entails:
	$\forall x\ \neg P(x) \Rightarrow R(x)$	$\forall x\ R(x) \Rightarrow S(x)$	$\forall x\ S(x)$
	no Horn form		

- **Need for complete inference rules**
  - Completeness theorem (Gödel, 1930)
    - There are complete inference rules for first-order logic
  - Resolution algorithm (Robinson, 1965)

©1996-2006 Michel Pasquier. All rights reserved

end

ARTIFICIAL INTELLIGENCE

CSC304
CPE406
SC430

School of Computer Engineering
Nanyang Technological University

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - Frames and Semantic / Conceptual Networks.
  - Managing Retractions, Assumptions, and Explanations.

©1996-2006 Michel Pasquier. All rights reserved

Forward-Chaining Production Systems

- **Forward-chaining system**
  - Assertions instead of queries
    - Inference generates new knowledge until a criterion is met
  - Appropriate for condition-action rules
    - i.e. add percepts to the KB ,then infer actions to perform
  - Theorem provers too generic
    - First-order logic w/ resolution $\Rightarrow$ huge branching factor
- **Typical features**
  - Rule memory (KB): sentences $p_1 \Lambda \ldots \Lambda\ p_m \Rightarrow act_1 \Lambda \ldots \Lambda\ act_n$
  - Working memory (WM): positive literals with no variables
  - 3-step inference: matching, conflict resolution, acting

©1996-2006 Michel Pasquier. All rights reserved

Production Rules and Inference

Rule memory:

$A(x) \wedge B(x) \wedge C(y) \Rightarrow$ add $D(x)$

$A(x) \wedge B(y) \wedge D(x) \Rightarrow$ add $E(x)$

$A(x) \wedge B(x) \wedge E(x) \Rightarrow$ delete $A(x)$

Working memory: [1]

A(1), A(2), B(2), B(3), B(4), C(5)

Inference:

$A(2) \wedge B(2) \wedge C(5) \Rightarrow$ add $D(2)$

Working memory: [2]

A(1), A(2), B(2), B(3), B(4), C(5), D(2)

Inference:

$A(2) \wedge B(2) \wedge C(5) \Rightarrow$ add $D(2)$

$A(2) \wedge B(2) \wedge D(2) \Rightarrow$ add $E(2)$

$A(2) \wedge B(3) \wedge \ldots$

Working memory: [3]

A(1), A(2), B(2), B(3), B(4), C(5), D(2), E(2)

Inference:

$A(2) \wedge B(2) \wedge C(5) \Rightarrow$ add $D(2)$

$A(2) \wedge B(2) \wedge D(2) \Rightarrow$ add $E(2)$

$A(2) \wedge B(2) \wedge E(2) \Rightarrow$ delete $A(2)$

©1996-2006 Michel Pasquier. All rights reserved

Example of Production System

- **Sorting a character string**
  - e.g. “cbaca” → “aabcc”

    3-char production rules:

    ba ⇒ ab (1)
    ca ⇒ ac (2)
    cb ⇒ bc (3)

#	Working memory	Conflict set	Rule fired
0	c<u>ba</u>ca	{ 3, 1, 2 }	1
1	<u>ca</u>bca	{ 2 }	2
2	acb<u>ca</u>	{ 3, 2 }	2
3	ac<u>ba</u>c	{ 3, 1 }	1
4	a<u>ca</u>bc	{ 2 }	2
5	aa<u>cb</u>c	{ 3 }	3
6	aabcc	{ }	HALT

©1996-2006 Michel Pasquier. All rights reserved

Conflict Resolution

- **Control strategy**

  Which of the matching rules should be fired?

  – None: execute systematically all rules.

  – No duplication: do not execute the same rule on the same arguments twice.

  – Recency: favour rules that refer to elements recently created in WM

  – Specificity:

  favour rules that are more specific (have more constraints).

  e.g. Mammal(x) $\Rightarrow$ add Legs(x,4)
  Mammal(x) Λ Human(x) $\Rightarrow$ add Legs(x,2)

  – Operation priority: favour rules that yield high-priority actions.

  e.g. Dusty(x) $\Rightarrow$ Action(Dust(x))
  Dangerous(x) $\Rightarrow$ Action(Leave(x))

©1996-2006 Michel Pasquier. All rights reserved

Using Production Systems

- **Forward-chaining production systems**
  - Modular systems
    - Inference engines (matching, conflict resolution, firing); OPS-5
    - Domain specific knowledge bases
  - Expert systems
    - Hundreds of commercial systems, from XCON (1982) onwards
    - Varied domains, such as accounting, biology, chemistry, computer eng., farming, finance, mathematics, medical diagnosis, etc.
  - Cognitive architectures
    - Models of human reasoning:

productions	long-term memory
working memory	short-term memory
new productions	learned knowledge

©1996-2006 Michel Pasquier. All rights reserved

A Simple Resolution

- **From propositional logic**
  - Unit Resolution
    - $$\frac{\alpha \vee \beta,\ \neg\beta}{\alpha}$$

    $(P \Rightarrow Q) \Leftrightarrow (\neg P \vee Q)$
  - Disjunctive Resolution
    - $$\frac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$$
  - Implicative Resolution
    - $$\frac{\neg\alpha \Rightarrow \beta,\ \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$
- **Interpretations**
  - Reasoning by case, i.e. given β, either α or γ is true
  - Transitivity of implication

©1996-2006 Michel Pasquier. All rights reserved

Generalised Resolution (CNF)

– Generalised Disjunctive Resolution

- $$\frac{\alpha_1 \vee \ldots\ \alpha_j \ldots \vee \alpha_M ,\ \ \gamma_1 \vee \ldots\ \gamma_k \ldots \vee \gamma_N}{\mathrm{SUBST}(\theta,\ \alpha_1 \vee \ldots\ \alpha_{j-1} \vee \alpha_{j+1} \ldots \vee \alpha_M \vee \gamma_1 \vee \ldots\ \gamma_{k-1} \vee \gamma_{k+1} \ldots \vee \gamma_N)} \qquad \mathrm{UNIFY}(\alpha_j, \neg\gamma_k) = \theta$$

– Examples

- $A \vee B,\ \neg B \vee C \quad \vdash \quad A \vee C$
- $A \vee B \vee \neg D,\ \neg B \vee C \quad \vdash \quad A \vee C \vee \neg D$
- $A \vee B \vee \neg D \vee E,\ C \vee D \vee \neg F \quad \vdash \quad A \vee B \vee C \vee E \vee \neg F$
- $A \vee B(K),\ \neg B(x) \vee C(x) \quad \vdash \quad A \vee C(K)$ under *{x/K}*
- $\neg A \vee \neg B(x) \vee E(x),\ B(y) \vee C(y) \quad \vdash \quad \neg A \vee C(x) \vee E(x)$
  under *{x/y}*

©1996-2006 Michel Pasquier. All rights reserved

Generalised Resolution (INF)

– Generalised Implicative Resolution

- $$\frac{\alpha_1 \Lambda \ldots \alpha_j \ldots \Lambda \alpha_{M1} \Rightarrow \rho_1 \vee \ldots \vee \rho_{M2}, \quad \sigma_1 \Lambda \ldots \Lambda \sigma_{N1} \Rightarrow \gamma_1 \vee \ldots \gamma_k \ldots \vee \gamma_{N2}}{\mathrm{SUBST}(\theta, \alpha_1 \Lambda \ldots \alpha_{j-1} \Lambda \alpha_{j+1} \ldots \Lambda \alpha_{M1} \Lambda \sigma_1 \Lambda \ldots \Lambda \sigma_{N1} \Rightarrow \rho_1 \vee \ldots \vee \rho_{M2} \vee \gamma_1 \vee \ldots \gamma_{k-1} \vee \gamma_{k+1} \ldots \vee \gamma_{N2})} \qquad \mathrm{UNIFY}(\alpha_j, \gamma_k) = \theta$$

– Examples

- $A \Rightarrow B,\ B \Rightarrow C \quad \vdash \quad A \Rightarrow C$
- $A \Lambda E \Rightarrow B,\ B \Rightarrow C \quad \vdash \quad A \Lambda E \Rightarrow C$
- $A \Rightarrow B,\ B \Lambda F \Rightarrow C \quad \vdash \quad A \Lambda F \Rightarrow C$
- $A \Rightarrow B,\ B \Rightarrow C \vee G \quad \vdash \quad A \Rightarrow C \vee G$
- $A \Rightarrow B \vee H,\ B \Rightarrow C \quad \vdash \quad A \Rightarrow C \vee H$
- $A \Rightarrow B(K),\ B(x) \Lambda F \Rightarrow C(x) \quad \vdash \quad A \Lambda F \Rightarrow C(K)$ under $\{x/K\}$

©1996-2006 Michel Pasquier. All rights reserved

Canonical Forms of Resolution

- **Conjunctive Normal Form (CNF)**
  - All sentences are a disjunction of literals, *negated or not:*
    $\alpha_1 \vee \ldots \vee \alpha_N$
- **Implicative Normal Form (INF)** or Kowalski form
  - All sentences are implications of *non-negated literals,* with a conjunction of premises and a disjunction of consequents: $\alpha_1 \wedge \ldots \wedge \alpha_M \Rightarrow \beta_1 \vee \ldots \vee \beta_N$
- **Conjunctive knowledge base**
  - All sentences joined in one big, implicit conjunction
    - e.g. P, $Q \Rightarrow R$, $\alpha \wedge \beta$, $\gamma_1 \vee \ldots \vee \gamma_N$ is equivalent to
      $(P) \wedge (Q \Rightarrow R) \wedge (\alpha \wedge \beta) \wedge (\gamma_1 \vee \ldots \vee \gamma_N)$

©1996-2006 Michel Pasquier. All rights reserved

Equivalence of CNF and INF

– Conversion

- $\beta_1 \vee \beta_2 \vee \neg\alpha_1 \vee \ldots \vee \beta_j \vee \ldots \vee \neg\alpha_k \vee \ldots \vee \beta_M \vee \ldots \vee \neg\alpha_N$ *CNF*

  $\neg\alpha_1 \vee \ldots \vee \neg\alpha_N \vee \beta_1 \vee \ldots \vee \beta_M$

  $\neg (\alpha_1 \wedge \ldots \wedge \alpha_N) \vee (\beta_1 \vee \ldots \vee \beta_M)$

  $\alpha_1 \wedge \ldots \wedge \alpha_N \Rightarrow \beta_1 \vee \ldots \vee \beta_M$ *INF*

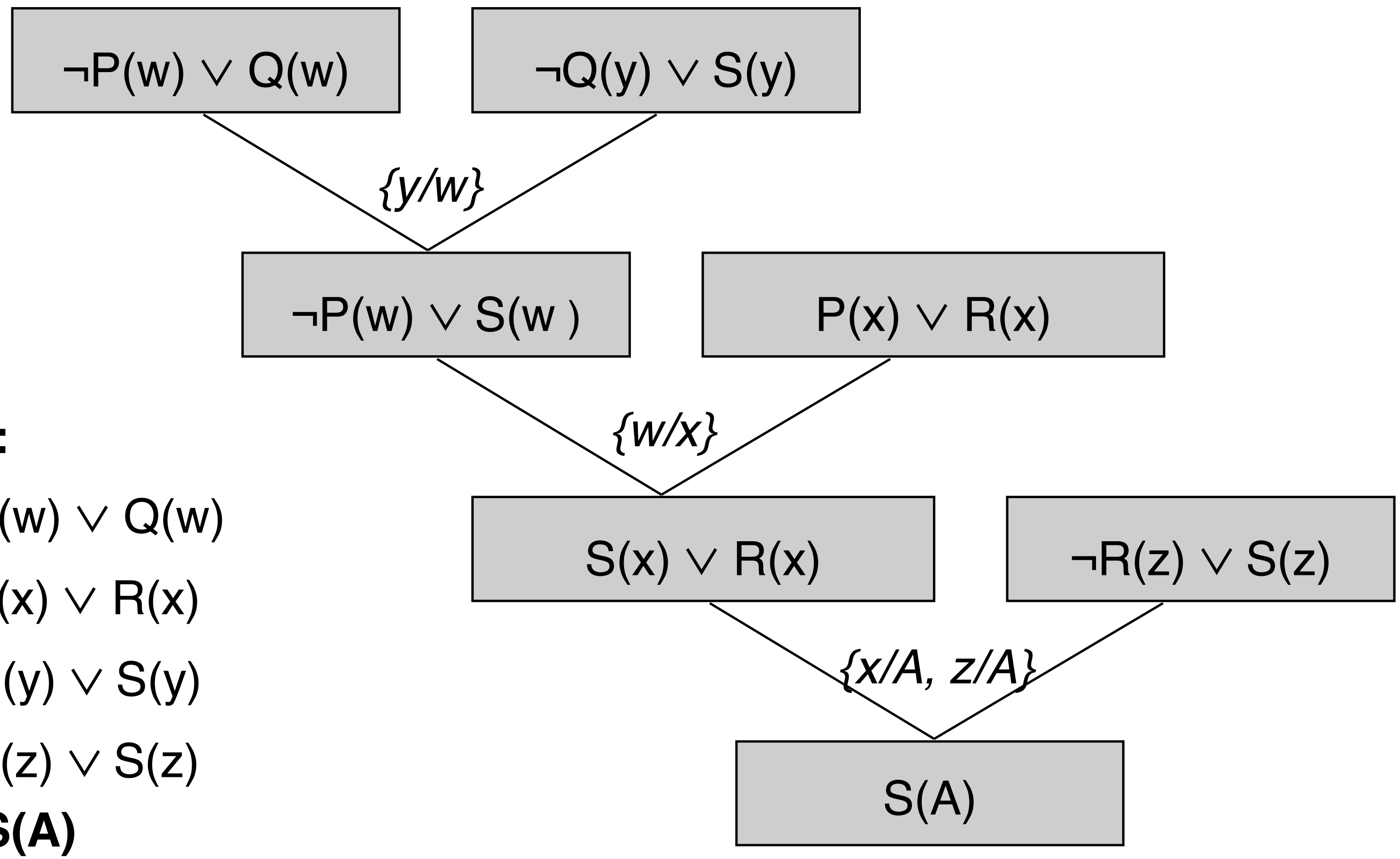
– Examples

- $A \vee B \Leftrightarrow \text{True} \Rightarrow A \vee B$
- $\neg A \vee B \Leftrightarrow A \Rightarrow B$
- $\neg A \vee \neg B \Leftrightarrow A \wedge B \Rightarrow \text{False}$
- $A \vee \neg B \vee \neg C \vee D \vee \neg E \Leftrightarrow B \wedge C \wedge E \Rightarrow A \wedge D$

©1996-2006 Michel Pasquier. All rights reserved

2ANIM

Example of Resolution Proof (CNF)



KB:

¬P(w) ∨ Q(w)

P(x) ∨ R(x)

¬Q(y) ∨ S(y)

¬R(z) ∨ S(z)

? S(A)

©1996-2006 Michel Pasquier. All rights reserved

Conversion to Normal Form

- **From first order logic to NF:**
  - Eliminate implications
    - i.e. $P \Rightarrow Q$ becomes $\neg P \vee Q$
  - Reduce scope of negations, using De Morgan's laws
    - i.e. $\neg(P \vee Q)$ becomes $\neg P \wedge \neg Q$,
      $\neg(P \wedge Q)$ becomes $\neg P \vee \neg Q$, $\neg\neg P$ becomes P
      $\neg\forall x\, P$ becomes $\exists x\, \neg P$, and $\neg\exists x\, \neg P$ becomes $\forall x\, P$
  - Standardise sentences apart, renaming variables
    - e.g. $(\forall x\, P(x)) \vee (\exists x\, Q(x))$ becomes $(\forall x\, P(x)) \vee (\exists y\, Q(y))$
  - Move quantifiers left
    - e.g. $P(x) \wedge (\forall y\, Q(y))$ becomes $\forall y\, P(x) \wedge Q(y)$ …

©1996-2006 Michel Pasquier. All rights reserved

Conversion to Normal Form (2)

- Remove existential quantifiers (Skolemization)
  - Replacing by a constant, e.g.: $\exists x\ P(x)$ becomes $P(C21)$
  - Replacing by a function, e.g.:
    $\forall x\ Person(x) \Rightarrow \exists y\ Mother(y, x)$, "Everyone has a mother"
    - w/ a constant: $\forall x\ Person(x) \Rightarrow Mother(Mum, x)$ → *wrong!*
    - w/ a Skolem function: $\forall x\ Person(x) \Rightarrow Mother(Mum(x), x)$
- Drop the universal quantifiers
- Distribute conjunctions ($\wedge$) over disjunctions ($\vee$)
  - e.g. $(P \wedge Q) \vee R$ becomes $(P \vee R) \wedge (Q \vee R)$
- Flatten nested conjunctions and disjunctions → **CNF**
  - e.g. $(P \vee Q) \vee R$ becomes $P \vee Q \vee R$
- Convert disjunctions back to implications → **INF**

©1996-2006 Michel Pasquier. All rights reserved

Summary

- **Inference rules for first-order logic …**
  - Are simply extended from propositional logic.
  - Are complex to use, because of a huge branching factor.
- **Unification …**
  - Improves efficiency by identifying appropriate variable substitutions.
- **The Generalised Modus Ponens …**
  - Uses unification to provide a powerful inference rule.
  - Can be either data-driven, using forward-chaining, or goal-oriented, using backward-chaining.

  …

©1996-2006 Michel Pasquier. All rights reserved

Summary

- Uses sentences in Horn form, i.e. $\alpha_1 \Lambda \ldots \Lambda \alpha_n \Rightarrow \beta$.
- Is not complete.

- **The Generalised Resolution …**
  - Provides a complete system for proof by refutation.
  - Requires sentences in either Conjunctive Normal Form or Implicative Normal Form, i.e. $\alpha_1 \Lambda \ldots \Lambda \alpha_m \Rightarrow \beta_1 \vee \ldots \vee \beta_n$ (which are equivalent).
  - Can use several strategies (heuristics) to improve efficiency and reduce the size of the search space.

©1996-2006 Michel Pasquier. All rights reserved

References

- Boolos, G. S. and Jeffrey, R. C. (1989). *Computability and Logic*. Cambridge University Press, Cambridge, third edition.
- Genesereth, M. R. and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, California.
- Hunter, G. (1971). *Metalogic: An Introduction to the Metatheory of Standard First-Order Logic*. University of California Press, Berkeley and Los Angeles.
- Siekmann, J. and Wrightson, G., editors (1983). *Automation of Reasoning*. Springer-Verlag, Berlin. Two volumes.
- Wos, L., Overbeek, R., Lusk, E., and Boyle, J. (1992). *Automated Reasoning: Introduction and Applications*. McGraw-Hill, New York, second edition.

©1996-2006 Michel Pasquier. All rights reserved

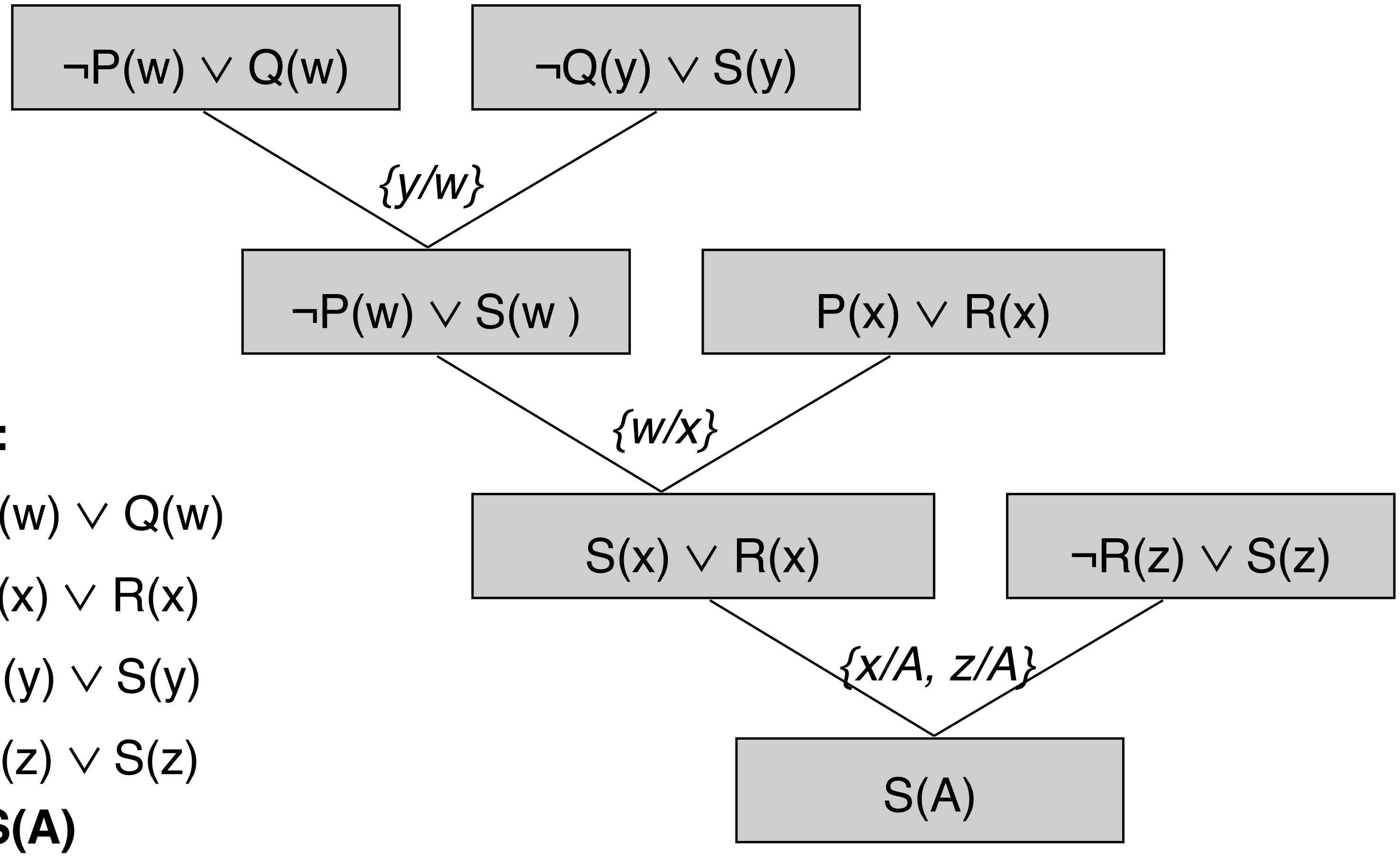
end

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - Frames and Semantic / Conceptual Networks.
  - Managing Retractions, Assumptions, and Explanations.

©1996-2006 Michel Pasquier. All rights reserved

Example of Resolution Proof (CNF)



KB:

$\neg P(w) \lor Q(w)$

$P(x) \lor R(x)$

$\neg Q(y) \lor S(y)$

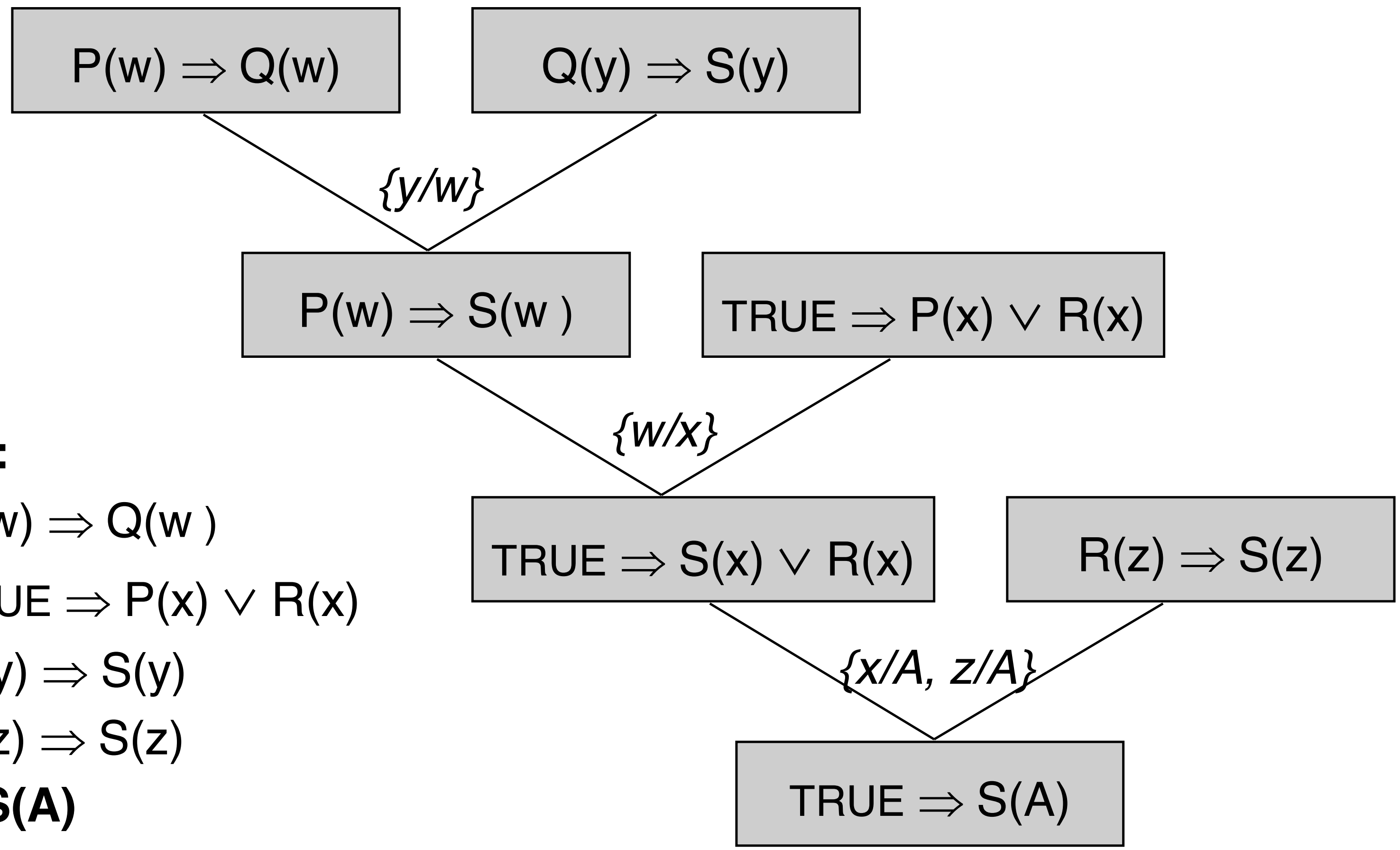
$\neg R(z) \lor S(z)$

? S(A)

©1996-2006 Michel Pasquier. All rights reserved

2ANIM

Example of Resolution Proof (INF)



KB:

$P(w) \Rightarrow Q(w)$

$TRUE \Rightarrow P(x) \vee R(x)$

$Q(y) \Rightarrow S(y)$

$R(z) \Rightarrow S(z)$

? S(A)

©1996-2006 Michel Pasquier. All rights reserved

Refutation

- **Resolution is incomplete**
  - There exist entailed sentences that it cannot prove
    - e.g. Empty KB (!), KB $\vdash$ $P \vee \neg P$, but resolution cannot prove it
- **Resolution by refutation**
  - Proof by contradiction (reductio ad absurdum):
    To prove P true, assume it false and prove a contradiction
    i.e.

    $$(KB \wedge \neg P \Rightarrow FALSE) \quad \Leftrightarrow \quad (KB \Rightarrow P)$$

  - Simple, sound, complete
    - e.g. assume $\neg(P \vee \neg P)$, rewrite as $\neg P \wedge P$, infer contradiction.

©1996-2006 Michel Pasquier. All rights reserved

Another Example Proof

- **Problem statement and query**
  - *"Jack owns a dog. Every dog owner is an animal lover. No animal lover kills an animal. Either Jack or curiosity killed the cat, who is named Tuna."*
  - *Did curiosity kill the cat?*
- **Translation in First Order Logic**
  - 1. $\exists x\ Dog(x) \land Owns(Jack,x)$
  - 2. $\forall x\ \forall y\ Owns(x,y) \land Dog(y) \Rightarrow AnimalLover(x)$
  - 3. $\forall x\ AnimalLover(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Kills(x,y))$
  - 4. $Kills(Jack,Tuna) \lor Kills(Curiosity,Tuna)$
  - 5. $Cat(Tuna)$
  - 6. $\forall x\ Cat(x) \Rightarrow Animal(x)$ [background knowledge]

©1996-2006 Michel Pasquier. All rights reserved

Variations on Translation to FOL

- **Example**
  - Statement: “*No animal lover kills an animal.*”
  - Translation in FOL:

1. $\forall x\ L(x) \Rightarrow (\forall y\ A(y) \Rightarrow \neg K(x,y))$

   $\forall x\ \neg L(x) \vee (\forall y\ \neg A(y) \vee \neg K(x,y))$

   $\forall x\ \forall y\ \ \neg L(x) \vee \neg A(y) \vee \neg K(x,y)$ **CNF**

2.

   $\forall x,y\ \neg(L(x) \wedge A(y)) \vee \neg K(x,y)$

   $\forall x,y\ \ L(x) \wedge A(y) \Rightarrow \neg K(x,y)$

3.

   $\neg\ (L(x) \wedge A(y) \wedge K(x,y))$

   $L(x) \wedge A(y) \wedge K(x,y) \Rightarrow F$

INF

©1996-2006 Michel Pasquier. All rights reserved

Conversion from FOL to CNF

- 1. $\exists x\ Dog(x) \land Owns(Jack,x)$
  - $\rightarrow$ 1a. $Dog(D)$
  - $\rightarrow$ 1b. $Owns(Jack,D)$
- 2. $\forall x\ \forall y\ Owns(x,y) \land Dog(y) \Rightarrow AnimalLover(x)$
  - $\rightarrow$ 2. $\neg Dog(y) \lor \neg Owns(x,y) \lor AnimalLover(x)$
- 3. $\forall x\ AnimalLover(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Kills(x,y))$
  - $\rightarrow$ 3. $\neg AnimalLover(u) \lor \neg Animal(v) \lor \neg Kills(u,v)$
- 4. $Kills(Jack,Tuna) \lor Kills(Curiosity,Tuna)$
  - $\rightarrow$ 4. $Kills(Jack,Tuna) \lor Kills(Curiosity,Tuna)$
- 5. $Cat(Tuna)$
  - $\rightarrow$ 5. $Cat(Tuna)$
- 6. $\forall x\ Cat(x) \Rightarrow Animal(x)$
  - $\rightarrow$ 6. $\neg Cat(z) \lor Animal(z)$

Query:

? $Kills(Curiosity,Tuna)$

$\neg Kills(Curiosity,Tuna)$

©1996-2006 Michel Pasquier. All rights reserved

Conversion from FOL to INF

- 1. $\exists x\ Dog(x) \land Owns(Jack,x)$
  - $\rightarrow$ 1a. $Dog(D)$
  - $\rightarrow$ 1b. $Owns(Jack,D)$
- 2. $\forall x\ \forall y\ Owns(x,y) \land Dog(y) \Rightarrow AnimalLover(x)$
  - $\rightarrow$ 2. $Dog(y) \land Owns(x,y) \Rightarrow AnimalLover(x)$
- 3. $\forall x\ AnimalLover(x) \Rightarrow (\forall y\ Animal(y) \Rightarrow \neg Kills(x,y))$
  - $\rightarrow$ 3. $AnimalLover(u) \land Animal(v) \land Kills(u,v) \Rightarrow FALSE$
- 4. $Kills(Jack,Tuna) \lor Kills(Curiosity,Tuna)$
  - $\rightarrow$ 4. $Kills(Jack,Tuna) \lor Kills(Curiosity,Tuna)$
- 5. $Cat(Tuna)$
  - $\rightarrow$ 5. $Cat(Tuna)$
- 6. $\forall x\ Cat(x) \Rightarrow Animal(x)$
  - $\rightarrow$ 6. $Cat(z) \Rightarrow Animal(z)$

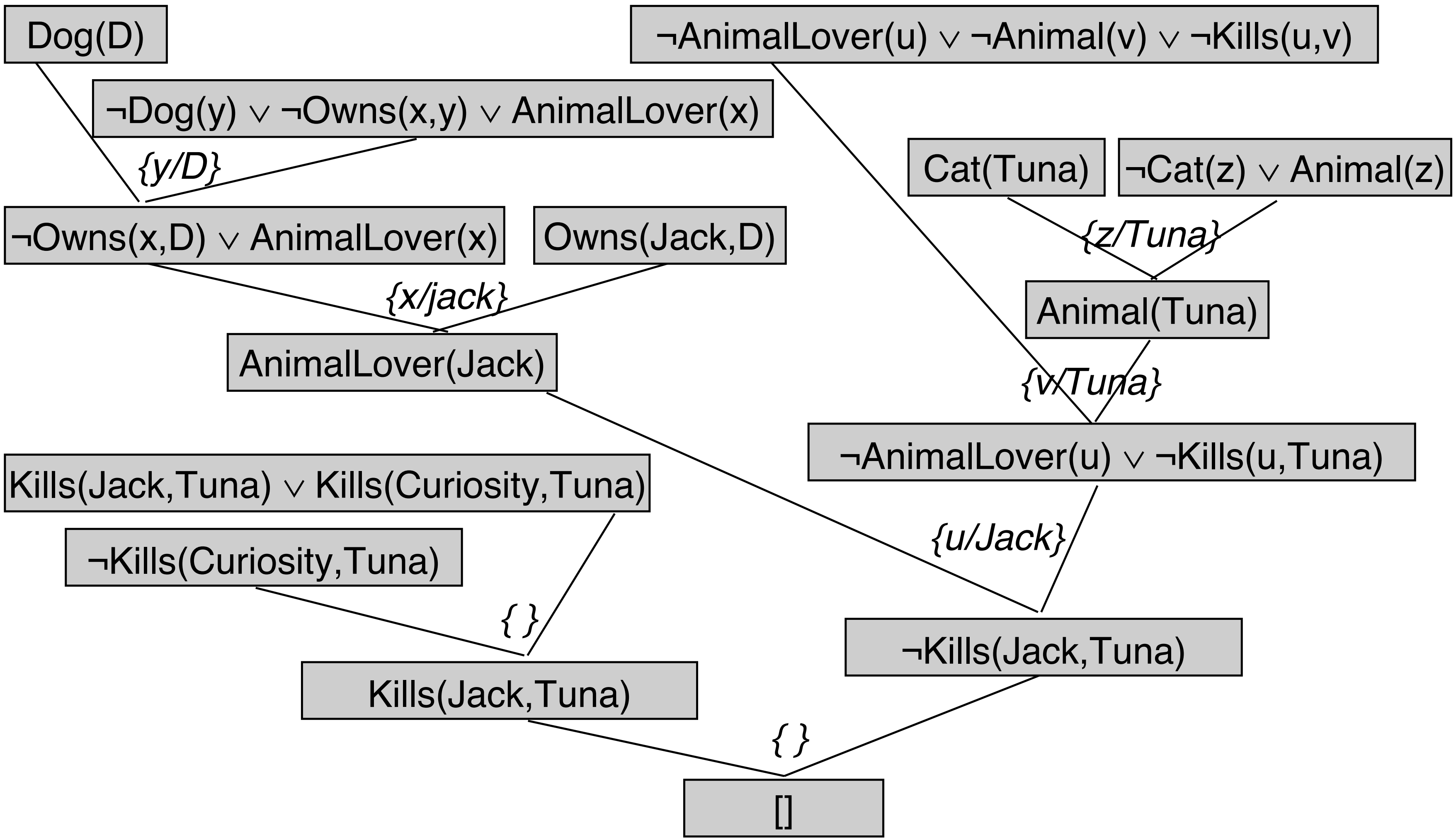
Query:

? $Kills(Curiosity,Tuna)$

$Kills(Curiosity,Tuna) \Rightarrow F$

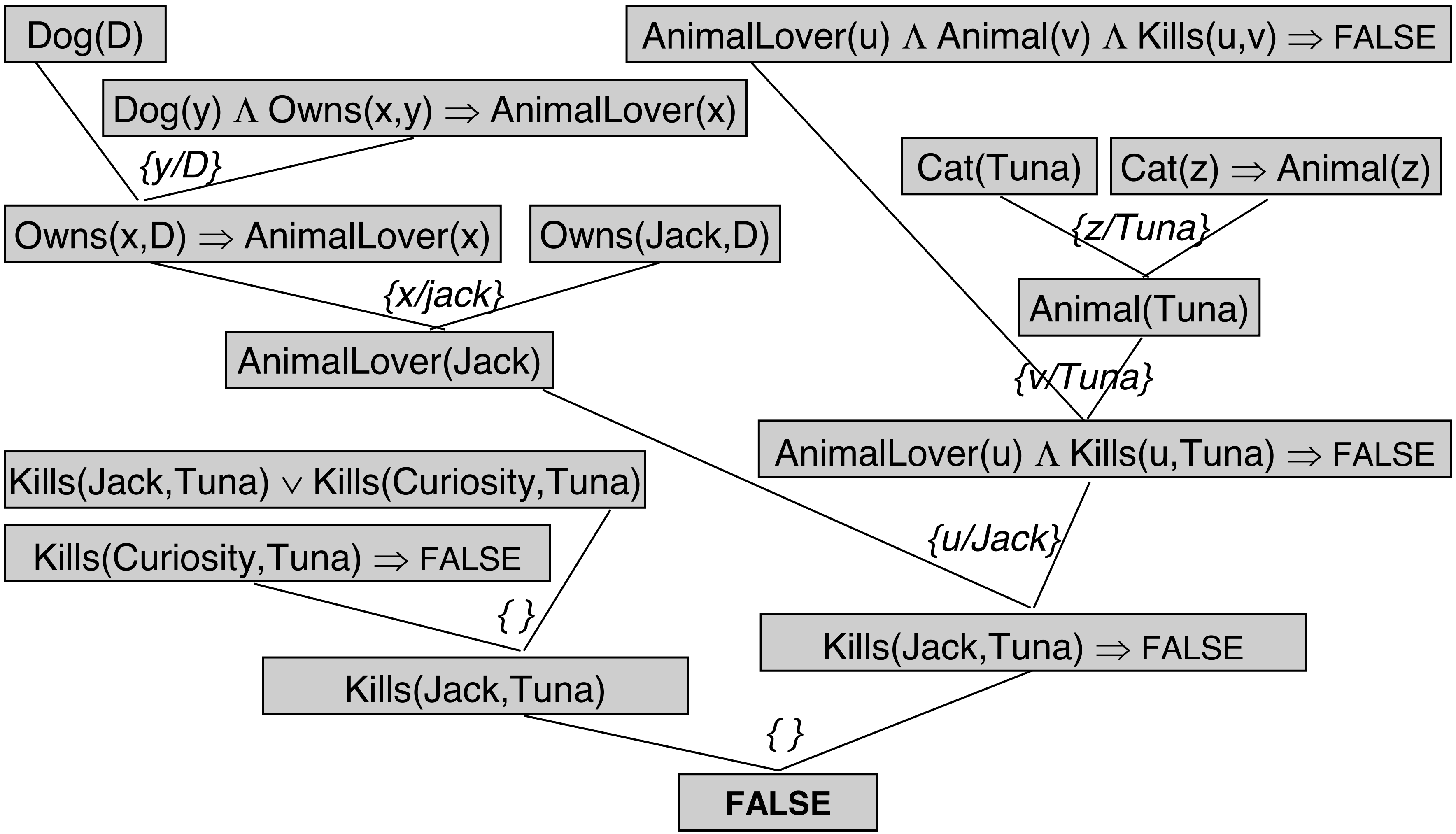
©1996-2006 Michel Pasquier. All rights reserved

Proof by Resolution–Refutation (CNF)



©1996-2006 Michel Pasquier. All rights reserved

Proof by Resolution–Refutation (INF)



©1996-2006 Michel Pasquier. All rights reserved

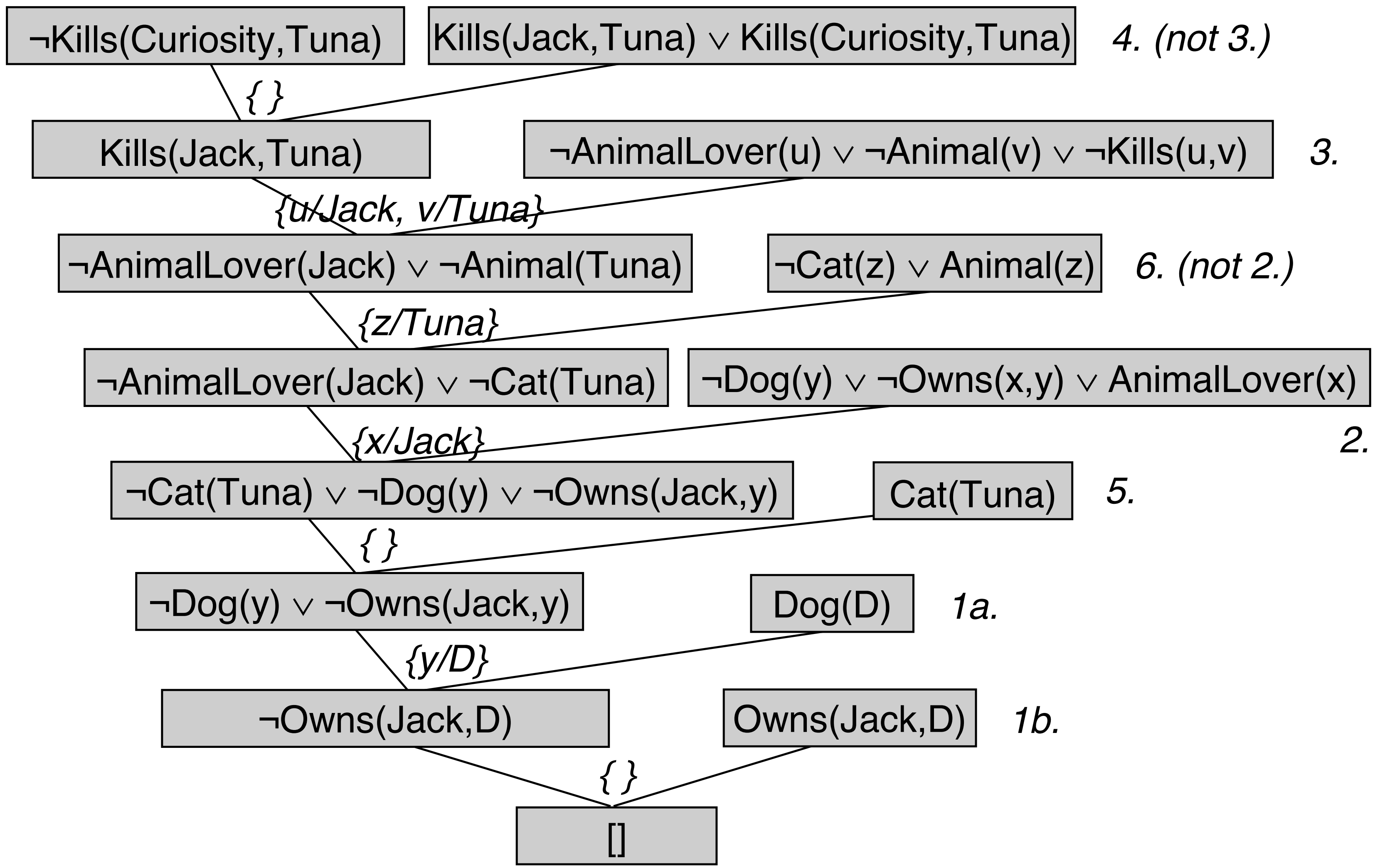
Resolution Strategies

- *Resolution is guaranteed to find a solution, but how?*

- **Unit Preference**
  - Favour short sentences (even unit clauses, if possible)
- **Set of Support**
  - Define a subset of the KB and use those sentences only
    - e.g. { negated query }, as in refutation
- **Input Resolution**
  - Combine KB sentences some inferred sentence
- **Subsumption**
  - Eliminate all sentences subsumed (more specific than) existing sentences in the KB, e.g. if P(x) then P(A) not needed

©1996-2006 Michel Pasquier. All rights reserved

Proof Using IR and UP Strategies



©1996-2006 Michel Pasquier. All rights reserved

Theorem Provers

- **Differences with LPL**
  - Use full first-order logic
  - Control kept distinct from knowledge
    - Order of writing does not matter, e.g. $A \Leftarrow B \wedge C$ or $A \Leftarrow C \wedge B$
- **Design of a theorem prover**
  - Design of a control strategy, e.g. OTTER
    - Using a set of support, unit preference, other strategies …
  - Extending Prolog , e.g. PTTP
    - Make search sound: use Occur-Check in unification
    - Make search complete: use IDS instead of depth-first search
    - Make inference complete: using linear input resolution
    - Use locking: store rules different ways
    - Allow negated literals

©1996-2006 Michel Pasquier. All rights reserved

Using Theorem Provers

- **Practical uses**
  - Assistant, e.g. decision making
  - Proof-checker, supervised by a mathematician
  - Socratic reasoner
    - Provide partial answers, so that a series of “right” questions always leads to the solution
  - Verification of software and hardware
    - e.g. computer algorithms (RSA, Boyer-Moore, etc.), logic design
  - Synthesis of software and hardware
    - i.e. prove “there exists a program p satisfying the specification s”
    - e.g. hand-guided synthesis of new algorithms,
      deductive synthesis in circuit design, large-scale integration

©1996-2006 Michel Pasquier. All rights reserved

Summary

- **Inference rules for first-order logic …**
  - Are simply extended from propositional logic.
  - Are complex to use, because of a huge branching factor.
- **Unification …**
  - Improves efficiency by identifying appropriate variable substitutions.
- **The Generalised Modus Ponens …**
  - Uses unification to provide a powerful inference rule.
  - Can be either data-driven, using forward-chaining, or goal-oriented, using backward-chaining.

  …

©1996-2006 Michel Pasquier. All rights reserved

Summary

- Uses sentences in Horn form, i.e. $\alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta$.
- Is not complete.

- **The Generalised Resolution …**
  - Provides a complete system for proof by refutation.
  - Requires sentences in either Conjunctive Normal Form or Implicative Normal Form, i.e. $\alpha_1 \wedge \ldots \wedge \alpha_m \Rightarrow \beta_1 \vee \ldots \vee \beta_n$ (which are equivalent).
  - Can use several strategies (heuristics) to improve efficiency and reduce the size of the search space.

©1996-2006 Michel Pasquier. All rights reserved

References

- Boolos, G. S. and Jeffrey, R. C. (1989). *Computability and Logic*. Cambridge University Press, Cambridge, third edition.
- Genesereth, M. R. and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, California.
- Hunter, G. (1971). *Metalogic: An Introduction to the Metatheory of Standard First-Order Logic*. University of California Press, Berkeley and Los Angeles.
- Siekmann, J. and Wrightson, G., editors (1983). *Automation of Reasoning*. Springer-Verlag, Berlin. Two volumes.
- Wos, L., Overbeek, R., Lusk, E., and Boyle, J. (1992). *Automated Reasoning: Introduction and Applications*. McGraw-Hill, New York, second edition.

©1996-2006 Michel Pasquier. All rights reserved

end