KNOWLEDGE REPRESENTATION

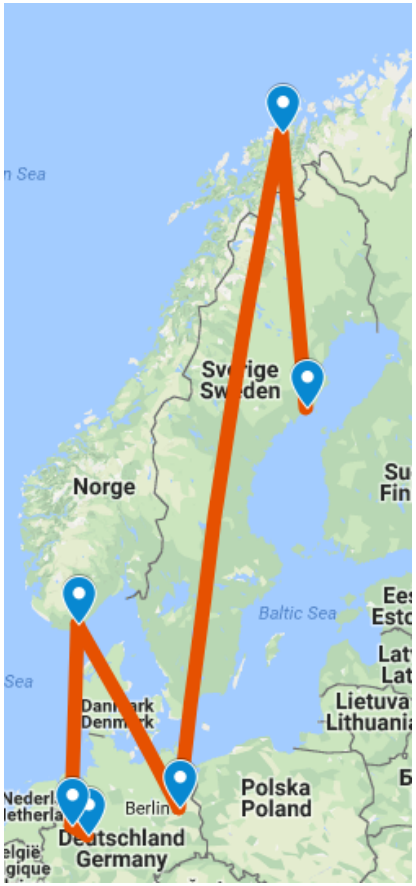
Chapter 7-7.5.1, 7.7.1, and 8- 8.2 (R & N)

Timotheus Kampik
(partly based on slides by Juan Carlos Nieves)

UMEÅ UNIVERSITY

THAT'S ME

- Bachelor's in Information Systems (Kristiansand)
- Master's Computer Science (Tromsø)
- Currently pursuing PhD at Umeå University:
  - WASP Sweden Autonomous Systems
- Industry background in
  - enterprise systems, product management
  - open source software
- Research interests:
  - AI and decision-making
  - Artificial intelligent agents of bounded rationality
  - Engineering multi-agent systems



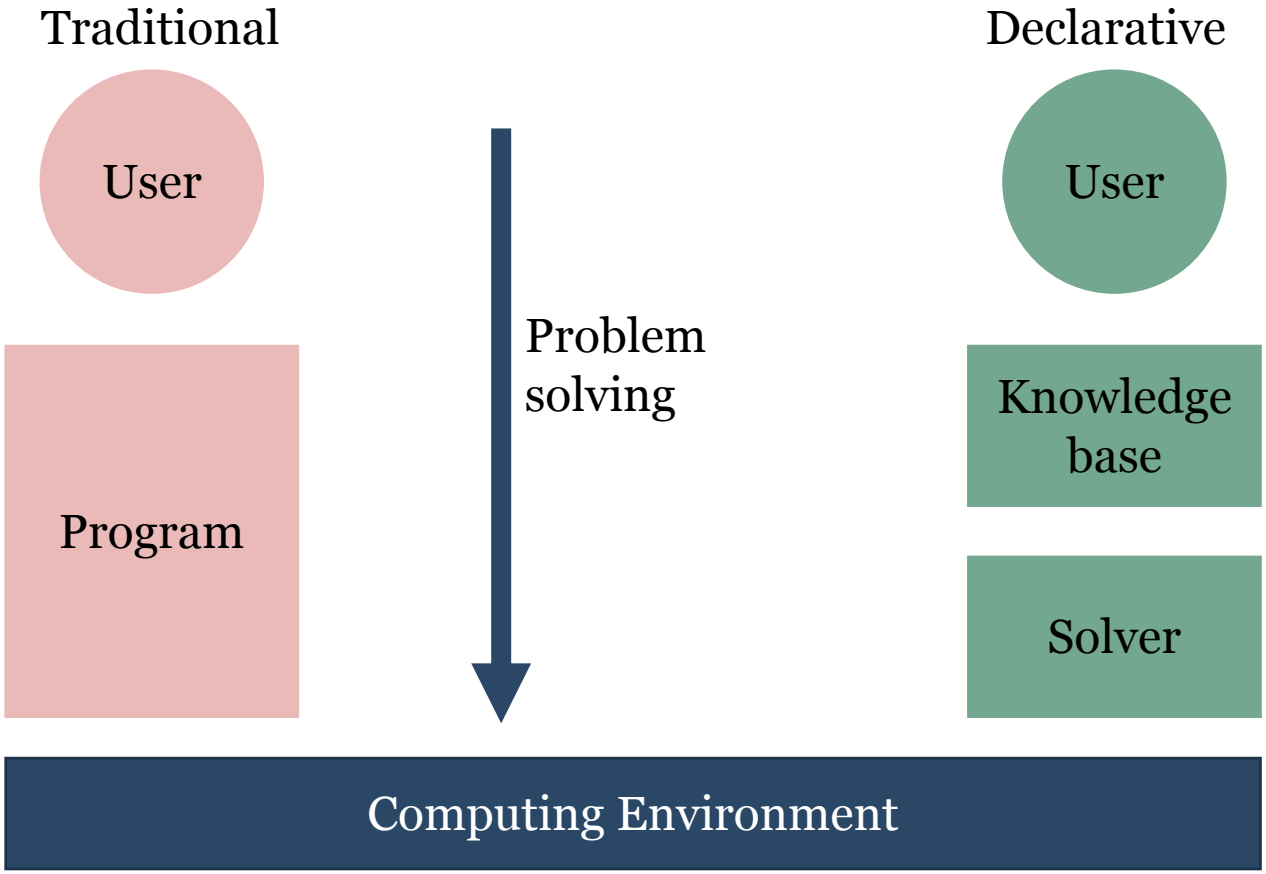
EXPECTED LEARNING OUTCOMES

- Recap: first-order logic
- Understand declarative programming and difference to “traditional” (imperative) programming
- Understand the difference between monotonic and non-monotonic reasoning
- Gain an intuition of how knowledge representation & reasoning can be applied in practice!

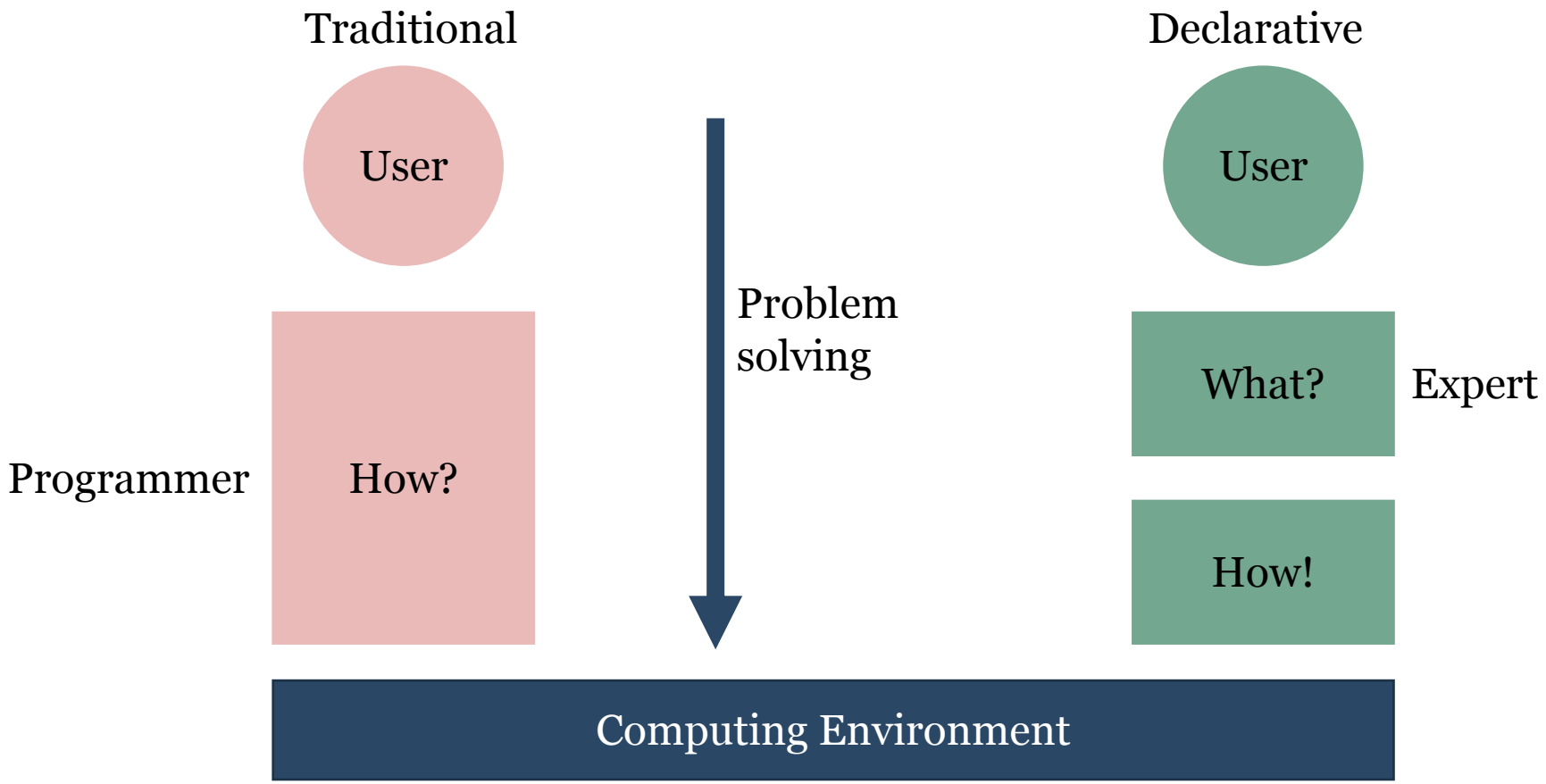
AGENDA

- Problem solving and knowledge-driven software
- Intuition: use cases
- Logic & First-Order Logic (FOL)
- Monotonic & non-monotonic reasoning
- Applications of knowledge representation & reasoning
    - Monitoring ML-based decision-making
    - First order logics and software development
    - Clinical decision-support

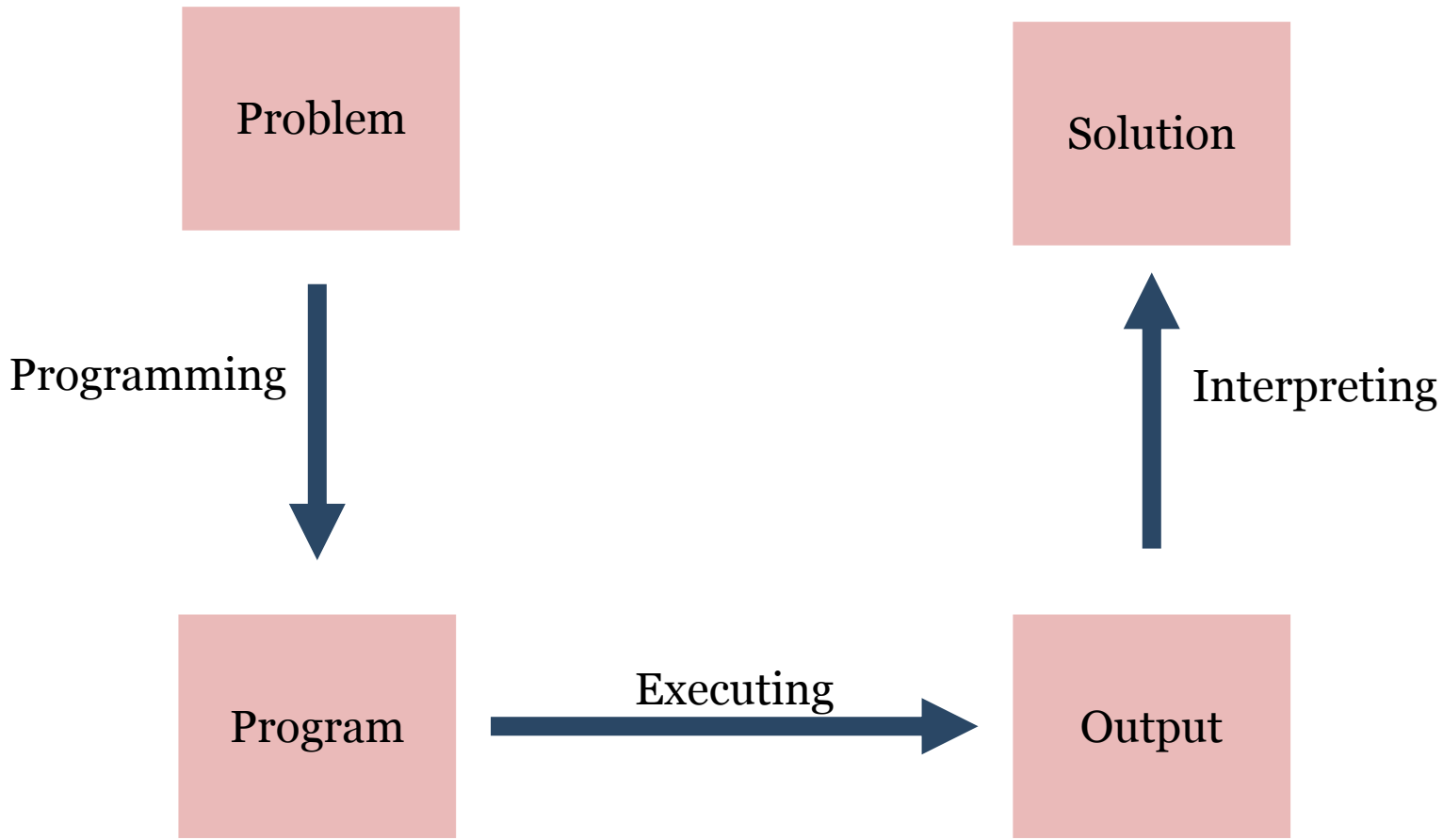
PROBLEM SOLVING & KNOWLEDGE DRIVEN SOFTWARE



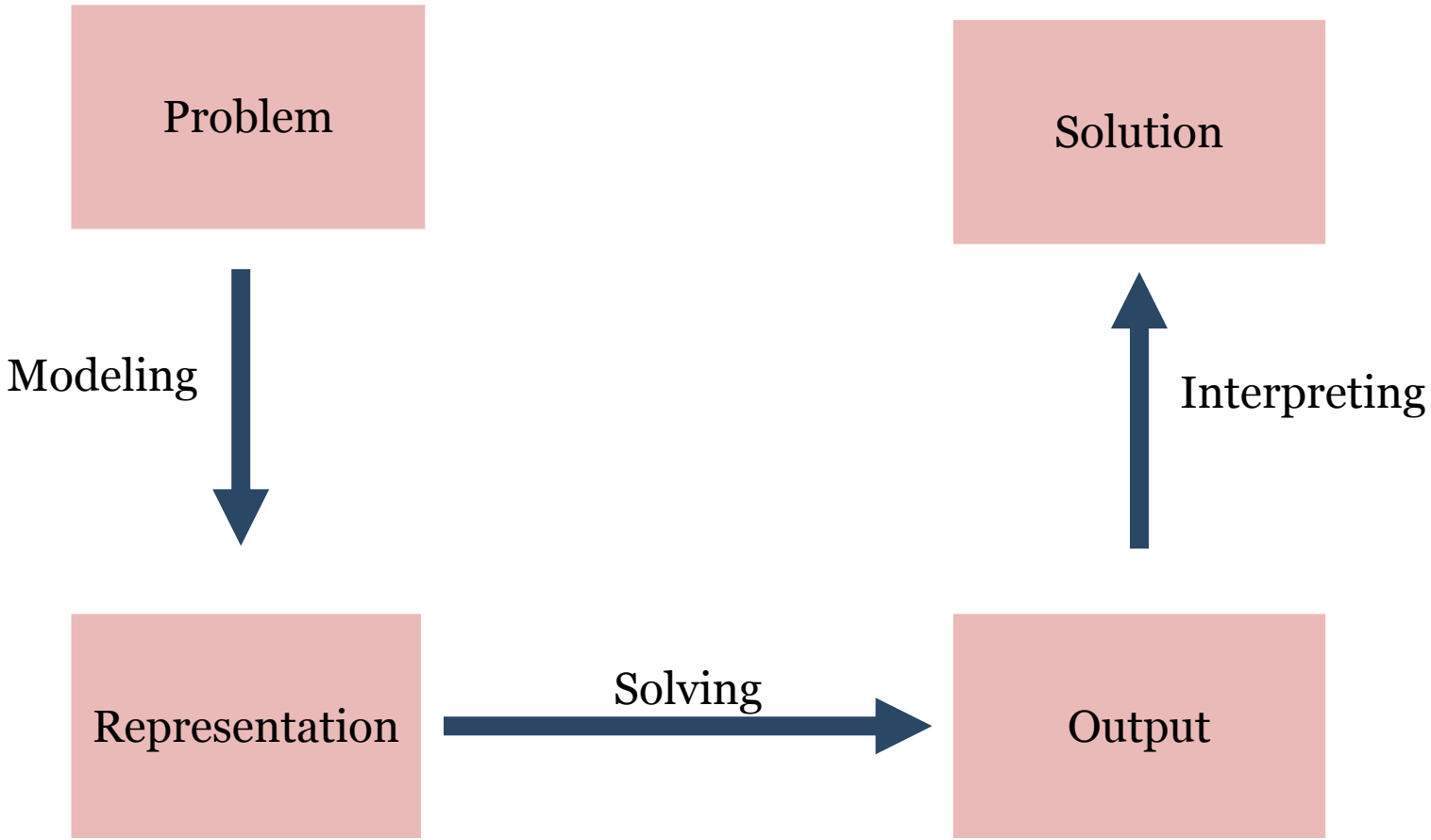
PROBLEM SOLVING & KNOWLEDGE DRIVEN SOFTWARE



TRADITIONAL PROGRAMMING



DECLARATIVE PROGRAMMING

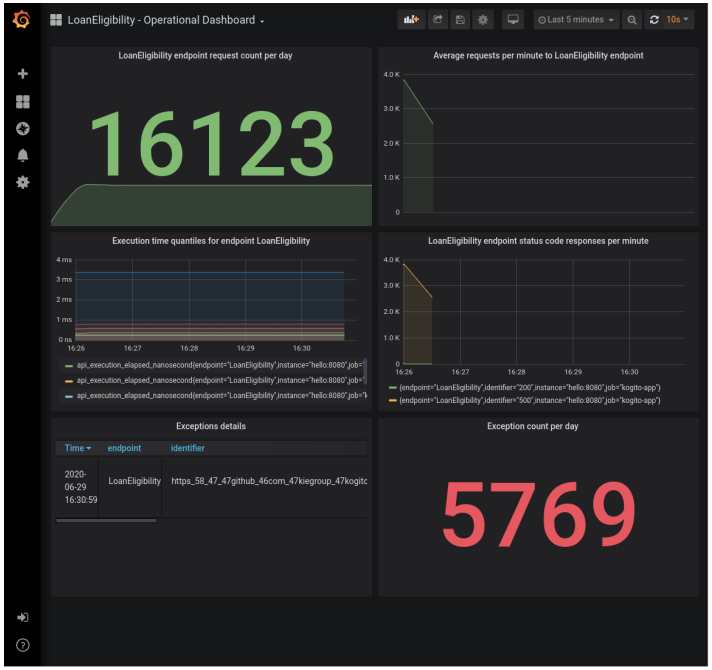


MOTIVATION: WHY KNOWLEDGE REPRESENTATION & REASONING?

Use Cases?

MOTIVATION: WHY DECLARATIVE PROGRAMMING?

Decision Monitoring



https://blog.kie.org/2020/06/trusty-ai-introduction.html
Rebecca Whitworth: ”TrustyAI introduction”
https://blog.kie.org/2020/07/trustyai-meets-kogito-decision-monitoring.html
Jacopo Rota. ”TrustyAI meets Kogito: decision monitoring

Clinical Decision-Support

https://www.redhat.com/files/resources/mi-rtinsights-northwell-analyst-paper-f12456-201805-en.pdf
redhat: ”Building a clinical decision engine platform with Red Hat Decision Manager

Reflect: when to use traditional programming, when declarative programming?

LOGICS AND FIRST-ORDER LOGIC

UMEÅ UNIVERSITY

LOGICS

- Logics are **formal languages** for representing **information** so that **conclusions** can be drawn
- A logics has:
  - **Syntax**: defines the sentences in the language/**grammar**
  - **Semantics**: defines the **meaning** of sentences or the truth of a sentence in a world

LOGICS

Example in arithmetic:

- `x + 2 > y` is a sentence
- `x + 2 >` is not a sentence
- `x + 2 > y` is true in a world where `x = 2, y = 1`
- `x + 2 > y` is false in a world where `x = 2, y = 8`

EXERCISE - LOGICS

Exercise, examples in arithmetic:

- Is `-(x + 2) > y` a sentence?
- Is `i++ == 2` a sentence?
- Is `i++ == 2` a sentence true in a world where i = 1?

LANDSCAPE OF LOGIC-BASED LANGUAGES: MONOTONIC VS. NON-MONOTONIC

- Monotonic: the truth of a proposition or axiom does not change when new information is added
  `All humans are mortal AND Socrates is a human ➔ Socrates is mortal`
- Non-monotonic: the truth of a proposition or axiom may change when new information is added
  `Birds typically fly AND Tweety is a bird ➔ Tweety presumably flies`

OPEN QUESTION: HOW TO ”REASONABLY” RESTRICT NON-MONOTONIC REASONING?

- See: *Gabbay. Theoretical Foundations for Non-Monotonic Reasoning in Expert Systems*
- Although the paper is from 1985, the question remains open

QUESTION TO YOU: CAN WE REASON NON-MONOTONICALLY WITH SIMPLE IF-CONDITIONS?

Example, JavaScript:

```
const rule = kb => {
      if (kb.isRaining === true)
      console.log("take umbrella")
}
```

LANDSCAPE OF LOGIC-BASED LANGUAGES

- Predicate logics (first order logic and its sub-languages)
  - Description logic
  - Web ontology logic
  - Extensions of FOL: Modal and epistemic logics
- Inconsistency-tolerant logic (non-monotonic)
- Representing vagueness
  - Probabilistic logics
  - Bayesian networks
  - Markov chains

FIRST-ORDER LOGIC (FOL)

- First-Order Logic (or: predicate logic) as a classical **monotonic** language is composed of
  1. Different types of **symbols**:
     - **Constants** or individuals of the world (upper case), e.g.:
       `John, 2, B Apple, Red, 9, Sara`
     - **Functions** that map individuals to individuals, e.g.:
       `motherOf(John) = Sara`
     - **Predicates** that map individuals to truth values, e.g.:
       `smaller(2, 9), color(Apple, Red)`

FIRST-ORDER LOGIC (FOL)

- First-Order Logic (or: predicate logic) as a classical **monotonic** language is composed of
  2. A set of **primitives**:
     - **Variables** (lower case), e.g.:
       `x, y, z`
     - **Connectives** (or logical operators):
       `¬, ∧,∨,⇔,⇒,(,)`
       Quantifiers:
       Universal (∀): ∀x, P: holds for all instantiations of P, e.g.:
       `∀x, (mother(x) ⇒ parent(x))`
       Existential (∃): ∃x, P: holds for some instantiations of P, e.g.:
       `∃x, (man(x) ∧ human(x))`

FIRST-ORDER LOGIC (FOL)

- First-Order Logic (or: predicate logic) as a classical **monotonic** language is composed of
  3. **Terms**:
     - Variables
     - Constants
     - n-ary functions applied to n arguments, e.g.: `a, x, f(a), g(x, b), f(g(x, g(b)))`
     - 0-ary function is a constant

FIRST-ORDER LOGIC (FOL)

- First-Order Logic (or: predicate logic) as a classical **monotonic** language is composed of
  4. **Atoms**: an n-ary predicate applied to n terms, e.g.: p(a)
  5. **Literals**: atoms or their negation, e.g.: `p(f(x), g(x, f(x))),¬p(f(x),¬g(x, f(x)))`
  6. **Formula** (or sentences):
     - Literal
     - Application of logical connectives to formula, e.g.: `p(f(x), x)⇒p(f(g(x, y))∧g(x, y))`

FOL SEMANTICS – TRUTH VALUES

- To find a meaning for a sentence (or formula) we need to calculate its truth value
- Each formula is evaluated to 2 truth values: true or false
- The truth value of a sentence depends on the truth values of the symbols in the sentence
- Determining the truth values of a formula is called interpretation
- Truth table enumeration is one way to calculate the truth value of a formula

FOL SEMANTICS – TRUTH VALUES

Truth table of the two symbol A and B, for all logical connectives:

A	B	¬ A	A ∧ B	A ∨ B	A ⇒ B	A ⇔ B
true	true	false	true	true	true	true
true	false	false	false	true	false	false
false	true	true	false	true	true	false
false	false	true	false	false	true	true

Given the "A" and "B" columns, you should be able to **compute (not memorize!)** the rest of the content of the table.

FOL SEMANTICS – INTERPRETATION

- An interpretation is an **assignment of meaning** to the symbols of a language.

- Logical formula that are syntactically correct do not have any meaning until they are given some interpretation.

- If a given interpretation assigns the value true to a formula, the interpretation is called a model of that formula.

FOL SEMANTICS – INTERPRETATION

An interpretation I: `(D,α)` consists of:

- Domain `D`: non-empty set of values
- Function α that assigns:
  - To each variable x, a value α[x] ∈ D
  - To each n-ary function f, a value in D:

$$\alpha[f] : D^n \rightarrow D$$

  - To each n-ary predicate P, either true or false:

$$\alpha[P] : D^n \rightarrow \{ true, false \}$$

FOL SEMANTICS – EXAMPLE I

$$F: q(x, y) \wedge p(x) \Rightarrow \neg p(y)$$

Interpretation I: (D, α), where:

- Domain $D = \{john = Father, sara = Mother, kate = Daughter, bob = Son\}$
- Function α:
  - for variables x and y :α[x] = john, α[y] = sara
  - for predicates p and q :-
    unary predicateα[p]: D1 → { true, false }
    p(x) returns true if x is male, otherwise false
    binary predicateα[q]: D2→{ true, false }
    q(x,y) returns true if x is sibling of y, otherwise false

The given formula F is true under I, and therefore I is a model of F

FOL SEMANTICS – EXAMPLE II

$$F: q(x, y) \wedge p(x) \Rightarrow \neg p(y)$$

Interpretation I: (D, α), where:

- Domain $D = \{john = Father, sara = Mother, tom = Son, bob = Son\}$
- Function α:
  - for variables x and y :α[x] = tom, α[y] = bob
  - for predicates p and q :-
    unary predicate α[p]: D1 → { true, false }
    p(x) returns true if x is male, otherwise false
    binary predicate α[q]: D2→{ true, false }
    q(x,y) returns true if x is sibling of y, otherwise false

The given formula F is false under I, and therefore I is not a model of F

FOL SEMANTICS – FORMULA

A formula is called:

- **Satisfiable** iff it is true for at least one interpretation. Therefore, the previous formula is satisfiable!
- **Valid** iff it is true for all interpretations
- **Unsatisfiable** iff it is not true for any interpretation

- Every interpretation that satisfies a formula is called a model of the formula

KNOWLEDGE BASE

A Knowledge Base (KB) is a set of sentences (formula) that are all connected with conjunction (∧), describing a situation in which the world is in.

Example:

- `Σ={ R=it is raining, W=streets are wet, D=it is dark, C=it is cold, H=John is home }`
- Then, a KB can be defined as:
  `KB ={ (R⇒W) ∧ (D∧C⇒H) ∧ (D) ∧ (¬H) }`
  translated to:
  `If it is raining then the streets are wet.`
  `If it is dark and cold then John is at home.`
  `It is dark.`
  `John is not home.`

ENTAILMENT

Entailment ($\vDash$) means that one thing follows from another:

Example: KB $\vDash \alpha$ means that the knowledge base KB entails the sentence α if α is true in all words in which KB is true.

EXERCISE

Use a truth table to figure out whether the statement `people are sad` is entailed from this KB:

- `If it is dark and raining, people are sad.`
- `If it is raining, then it is dark.`
- `It is raining.`

REASONING

REASONING

- Reasoning or *inference* is the process of answering a question:
  $KB \vdash_p \alpha$ means α can be derived from KB by procedure p.

- Consequences of KB are a haystack, α is a needle
  - **Entailment**: needle is in the haystack
  - **Inference**: finding the needle, e.g. by truth table enumeration

REASONING

- Formal properties of a reasoning process:
  - **Soundness**: procedure p is sound iff:
    $(KB \vdash_p \alpha) \Rightarrow (KB \vDash \alpha)$
    KB entails everything that p infers
  - **Completeness**: procedure p is complete iff:
    $(KB \vDash \alpha) \Rightarrow (KB \vdash_p \alpha)$
    p infers everything that KB entails

COMPLETENESS ”IN THE WILD”

- Important for managing “business rules” (decision-support and automation systems)
- Identify missing rules
- Merge overlapping rules
- See: *Calvanese et al. Semantics, Analysis and Simplification of DMN Decision Tables*

COMPLETENESS ”IN THE WILD”

Decision logic: Check applicant

Decision Table
Expression
Link
Informal Expression

Add rule | Remove rule | Add input | Remove column | Errors detected | Import/Export

	Inputs				Outputs	Annotations
U	...tter of Application		Evaluate overall skills		Status	New Annotation
	{complete,incomplete}		{Acceptable,Unacceptable}		{Accept,Reject}	Informational
1	=	complete	=	Acceptable	Accept	The candidate is worthy of an interview
2	=	incomplete	=	Acceptable	Reject	
3	=	complete	=	Unacceptable	Reject	
+	Add new row					

Check Decision Table

#	Rule	Description
1	---	No rule exists for (incomplete, !Acceptable)

Cancel | Apply

UMEÅ UNIVERSITY

REASONING METHODS

Divide into (roughly)

- Model checking
  - Models are assignments of true and false to every symbol
  - Truth table enumeration
- Application of inference rules
  - Legitimate (sound) generation of new sentences from old
  - Proof = a sequence of inference rule applications
  - Typically require translation of sentences into a normal form
- Reasoning can be monotonic or non-monotonic!

APPLICATIONS

EXERCISE - APPLICATIONS

Think about some applications of

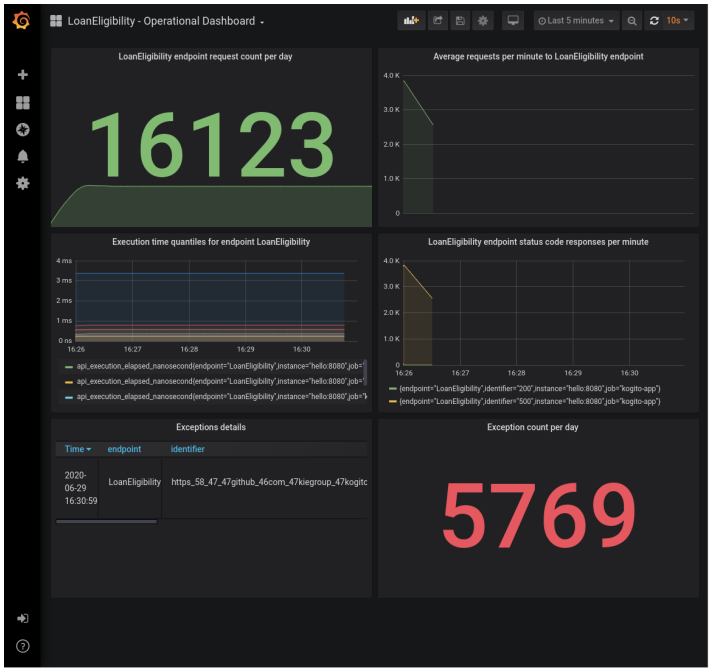
- Declarative programming
- First-order logic
- Non-monotonic reasoning

DECLARATIVE PROGRAMMING: DECISION MONITORING OF MACHINE LEARNING-BASED SYSTEMS

- Machine learning increasingly used for decision-automation
- Still, compliance with rules and regulations need to be checked or be guaranteed “by design”

➔Use declarative, rule-based systems for compliance checks

➔“Learn” human interpretable rules



https://blog.kie.org/2020/06/trusty-ai-introduction.html
Rebecca Whitworth: ”TrustyAI introduction”
https://blog.kie.org/2020/07/trustyai-meets-kogito-decision-monitoring.html
Jacopo Rota. ”TrustyAI meets Kogito: decision monitoring

FIRST ORDER LOGIC: RELEVANCE FOR SOFTWARE ENGINEERING

If-conditions and test assertions resemble first order logic

➔Understanding of first order logic a crucial part of a programmer's tool box

```
@[test]
+!test_egreedy_bandit_api :
    true
    <-
    //.init_bandit(tArmNames, type = "EPSILON_GREEDY", epsilon,
    decayRate, seed, handoverArg, unifier)
    .init_bandit([a, b, c, d, e], "EPSILON_GREEDY", 0.1, 0.1, 20, B);
    !assert_equals(0, B);
    .pull_arm(B, A);
    !assert_equals("e", A);
    .set_reward(B, A, 2, F);
    !assert_equals([0, 0, 0, 0, 1], F);
.
```

NON-MONOTONIC REASONING: RELEVANCE FOR DECISION SUPPORT & AUTOMATION

- Most large organizations use complex, rule-based systems
- Again, rules are necessary for compliance and quality assurance
- Non-monotonic reasoning (AKA: defeasible reasoning) allows for a structured approach towards managing rule conflicts

https://www.redhat.com/files/resources/mi-rtinsights-northwell-analyst-paper-f12456-201805-en.pdf
redhat: "Building a clinical decision engine platform with Red Hat Decision Manager

RECAP – LEARNING GOALS

EXPECTED LEARNING OUTCOMES

- **Recap: first-order logic**
- **Understand declarative programming and difference to “traditional” (imperative) programming**
- **Understand the difference between monotonic and non-monotonic reasoning**
- **Gain an intuition of how knowledge representation can be applied in practice!**

FURTHER READING

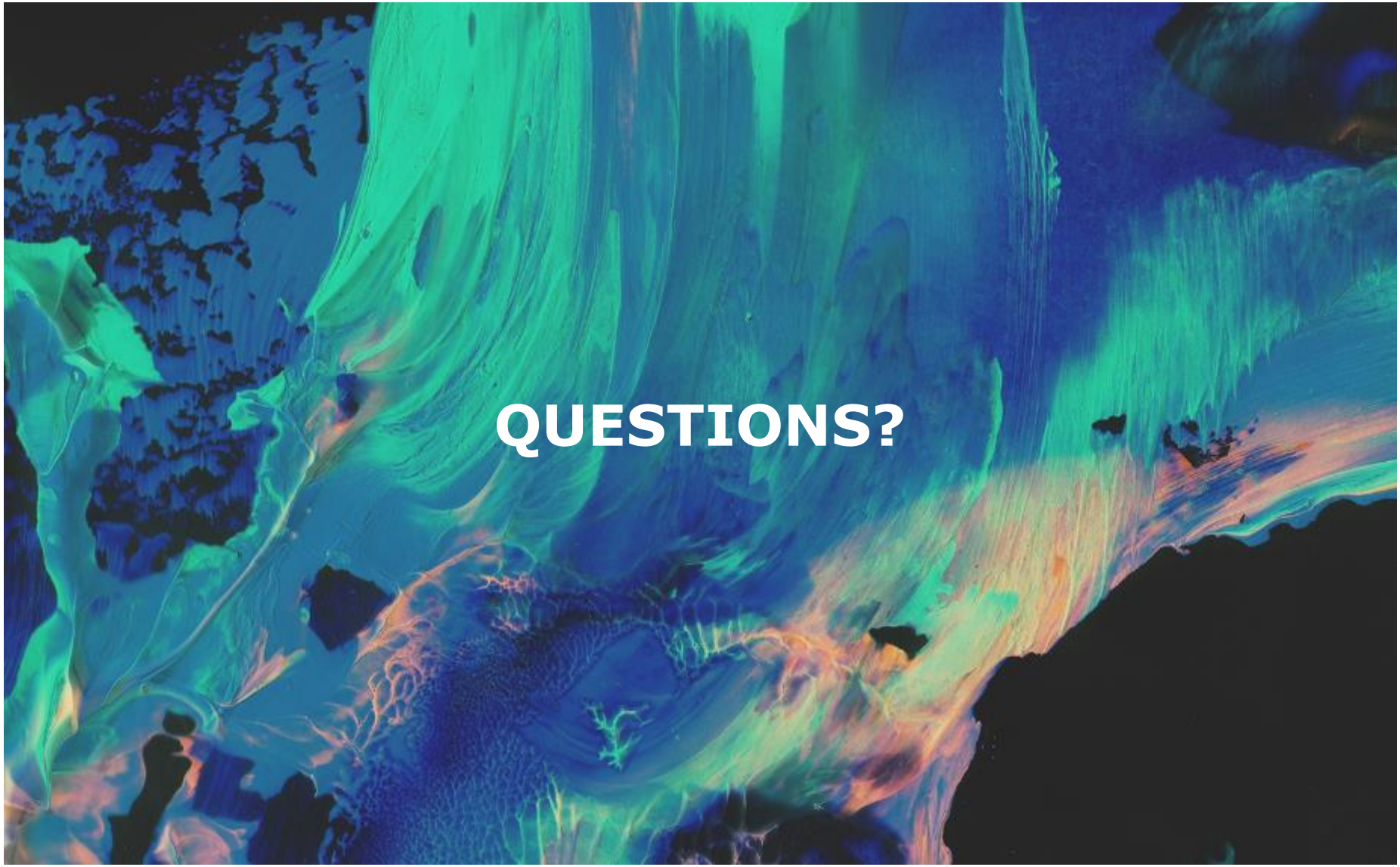
- *Russel, Norvig: Artificial Intelligence: A Modern Approach, chapters* 7-7.5.1, 7.7.1, and 8-8.2

Plus (optional, but recommended to take a look):

- *Gabbay. Theoretical Foundations for Non-Monotonic Reasoning in Expert Systems*
- *Calvanese et al. Semantics, Analysis and Simplification of DMN Decision Tables*

FURTHER CODING

- *Drools, industry-scale rules engine for declarative programming:* https://docs.jboss.org/drools/release/5.3.0.Final/drools-expert-docs/html/ch01.html



UMEÅ UNIVERSITY