Requirements Elaboration: An Inductive Search Problem

Dalal Alrajeh
Department of Computing
Imperial College London

The 25th CREST Open Workshop: Requirements and Test Optimisation
February 12th, 2013
Requirements

Sensors

Controller

Pump

HighWaterDetected

PumpOnWhenHighWaterDetected

PumpOnWhenHighWater

Positive scenarios
Fixing the Terms

• Goals
  Objectives to be achieved by system through agent cooperation

• Domain Properties
  Descriptive statements about the system (e.g., effect of operations)

• Scenarios
  How the software-to-be and its environment should and should not interact (positive and negative)

• Operational Requirements
  Constraints on the actions an agent can perform to achieve the
Requirements to Operational Requirements

Operational requirements on software agents

- $(\text{CriticalMethane} \implies \neg \text{switchPumpOn})$
- $((\text{CriticalMethane} \land \text{PumpOn}) \implies \neg \text{switchPumpOff})$
- $(\text{HighWater} \land \neg \text{CriticalMethane} \land \neg \text{PumpOn} \implies \text{switchPumpOn})$
- $((\text{HighWater} \land \neg \text{CriticalMethane}) \implies \neg \text{switchPumpOff})$
- $(\text{LowWater} \implies \neg \text{switchPumpOn})$
- $((\text{LowWater} \land \text{PumpOn}) \implies \neg \text{switchPumpOff})$
- $((\text{CriticalMethane} \land \neg \text{Alarm}) \implies \text{raiseAlarm})$
- $((\neg \text{CriticalMethane} \land \text{Alarm}) \implies \text{stopAlarm})$
Operational Requirements Elaboration

Domain Properties Goals

Negative Scenarios
Positive Scenarios

Elaborate

Operational requirements

Automated
Gives Alternatives

Sound
Consistent and Correct
Inductive Learning (1)

Given:
- $B$ (background theory)
- $E^+$ (set of positive examples)
- $E^-$ (set of negative examples)
- $IC$ (set of integrity constraints)

Find:
- $H$ (hypothesis such that)
  - $B \cup H \models E^+$
  - $B \cup H \not\models E^-$
  - $B \cup H \cup IC$ is consistent

Theory Completion
The secret of flight

Given:

\[ B = \{ \text{animal}(X). \text{bird}(X). \text{robin}(a). \text{pigeon}(b). \} \]
\[ E^+ = \{ \text{fly}(a). \text{fly}(b). \} \]
\[ E^- = \{ \} \]
\[ C = \{ \bot \leftarrow \text{bird}(X), \text{not animal}(X). \} \]

Find:

\[ H_1 = \{ \text{fly}(X) \leftarrow \text{bird}(X). \} \]
\[ H_2 = \{ \text{fly}(X) \leftarrow \text{animal}(X). \} \]
\[ H_3 = \{ \text{fly}(X). \} \]

Search space includes rules with the predicate \text{fly} in the head and the predicates \text{bird} and \text{penguin} in the body.

Optimal Solution is the smallest hypothesis that may be constructed
Inductive Learning (2)

Given:
- \( B \) (background theory)
- \( R \) (revisable theory)
- \( E^+ \) (set of positive examples)
- \( E^- \) (set of negative examples)
- \( IC \) (set of integrity constraints)

Find:
- \( R' \) (revised theory such that)
  - \( B \cup R' \models E^+ \)
  - \( B \cup R' \not\models E^- \)
  - \( B \cup R' \) is consistent
  - \( c(R, R') \) is minimal

Theory Revision
The secret of flight

Given:

\[ B = \{ \text{animal}(X). \text{bird}(X). \text{robin}(A). \text{pigeon}(b). \text{penguin}(c). \} \]
\[ E^+ = \{ \text{fly}(a). \text{fly}(b). \} \]
\[ E^- = \{ \text{fly}(c). \} \]
\[ IC = \{ \bot \leftarrow \text{bird}(X), \text{not animal}(X). \} \]
\[ R = \{ \text{fly}(X). \} \]

Find:

\[ H_1 = \{ \text{fly}(X) \leftarrow \text{not penguin}(X). \} \]

\[ c(R, R') = 1 \]

Optimal Solution is the hypothesis with minimal changes min(c(R, R')).

Search space includes rules with the predicate fly in the head and the predicates bird and penguin in the body.
Elaboration & Learning: Symmetrical?

Requirements
Elaboration

Negative Scenarios → Elaborate → Operational requirements
Positive Scenarios

Inductive Learning

Positive Examples → Inductive Learning → Hypothesis
Negative Examples

Domain Properties, Goals

Operational requirements

Background Theory, Constraints
Learning Operational Requirements

Learning Task

Given
A set of domain properties $D$, a partial set of operational requirements $O$ and scenarios $(S^+ U S^-)$ such that

$D U O \not\models s^+$ for some $s^+$ in $S^+$
$D U O \models s^-$ for some $s^-$ in $S^-$
$D U O U G$ is consistent

Find
The smallest set of operational requirements $O^*$ such that

$D U O U O^* \models s^+$ for all $s^+$ in $S^+$
$D U O U O^* \not\models s^-$ for all $s^-$ in $S^-$
$D U O U O^* U G$ is consistent

where $\models$ is interpreted as the linear temporal logic satisfaction relation with respect to traces in the semantic model (i.e. an LTS of $(D U O U O^*)$).
Mine Pump Example

The controller of a mine pump is expected to monitor and control water levels in a mine, to prevent water overflow. It is composed of a pump for pumping mine-water up to the surface and sensors for monitoring the water levels and methane percentage.

The pump must be activated once the water has reached preset high water level and deactivated once it reaches low water level.

Moreover, the pump must be switched off if the percentage of methane in the mine exceeds a certain critical limit.
Mine Pump Example

Goal: Achieve\([\text{PumpOnWhenHighWaterAndNoMethane}]\)
Informal Definition: The pump shall be on when the water level is above high water level and there is no methane present in the mine
Formal Definition
\[(\text{SYN}) \quad \square((\text{HighWater} \land \neg \text{CriticalMethane}) \rightarrow \lozenge \text{PumpOn})\]

Domain Property:
Operation: switchPumpOn
DomPre: \neg \text{PumpOn}   DomPost: \text{PumpOn}
Operation: switchPumpOff
DomPre: \text{PumpOn}   DomPost: \neg \text{PumpOn}
Operation: aboveHigh
DomPre: \neg \text{HighWater}   DomPost: \text{HighWater}

Positive Example

Negative Example
Coverage Check

Domain Property:
Operation: switchPumpOn
DomPre: ¬PumpOn  DomPost: PumpOn
Operation: switchPumpOff
DomPre: PumpOn  DomPost: ¬PumpOn
Operation: aboveHigh
DomPre: ¬ HighWater  DomPost: HighWater

Operational Requirement:

Domain Property:
Operation: switchPumpOn
DomPre: ¬PumpOn  DomPost: PumpOn
Operation: switchPumpOff
DomPre: PumpOn  DomPost: ¬PumpOn
Operation: aboveHigh
DomPre: ¬ HighWater  DomPost: HighWater

Operational Requirement:
Coverage Check

- Controller
  - Sensors
  - Pump
  - Switch Pump On
  - Switch Pump Off

- Sensors
  - Above Low
  - Above High

- Pump
  - Switch Pump On
  - Switch Pump Off

- States:
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7

- Transitions:
  - Switch Pump On
  - Switch Pump Off
  - Critical Methane
  - No Critical Methane

- Scenarios:
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7

- Conditions:
  - Above Low
  - Above High

- Coverage Check:
  - ⊨
  - ⊭
Learned Requirements

Domain Property:
Operation: `switchPumpOn`
DomPre: ¬PumpOn  DomPost: PumpOn
Operation: `switchPumpOff`
DomPre: PumpOn  DomPost: ¬PumpOn
Operation: `aboveHigh`
DomPre: ¬ HighWater  DomPost: HighWater

Operational Requirement:
Operation: `switchPumpOff`

\( H_1 = \text{ReqPre: } \text{HighWater} \land \neg \text{CriticalMethane} \)
\( (\text{HighWater} \land \neg \text{CriticalMethane}) \rightarrow \Box \neg \text{switchPumpOff} \)

\( H_2 = \text{ReqPre: } \text{HighWater} \)
\( ((\text{HighWater}) \rightarrow \Box \neg \text{switchPumpOff}) \)

\( H_3 = \text{ReqPre: } \neg \text{CriticalMethane} \)
\( (\neg \text{CriticalMethane}) \rightarrow \Box \neg \text{switchPumpOff} \)

\( H_4 = \text{ReqPre: } \{\} \)
\( (\Box \neg \text{switchPumpOff}) \)

Learning method returns minimal solution that would prevent s- from occurring
Minimal == Optimal?

Never switch the pump off
Requirement Optimisation

- Requirements are as good as the scenarios given
- Coverage of positive scenarios => Finding the minimal set of requirements that need to be true for the scenario to occur
- Coverage of negative scenarios => Finding the minimal set of requirements that need to be true for the scenario not to occur
- More positive scenarios => less restrictive requirements
Test cases & Learning: Symmetrical?

Test Case Generation

- (Negated) Specification
- ?
- Generate
- Program
- Constraints
- Test Case

Inductive Learning

- Positive Examples
- Negative Examples
- Inductive Learning
- Background Theory
- Constraints
- Hypothesis
Test Case Generation

Given
A program $P$, a set of constraints $C$ and a specification $S$ such that

\[ P \cup S \cup C \]

is consistent

Find
The smallest Test Case $T$ such that

\[ P \cup T \neq s \quad \text{for some } s \text{ in } S \]

\[ P \cup S \cup C \cup T \]

is consistent
Controlling the Search for an Optimal Solution

- How to reduce the hypothesis search space?
  - Language bias (e.g. Mode declarations, Occam’s razor principle)
  - Search bias (e.g. bottom-up, top-down)

- How much of the domain to capture in $B$?

- How to obtain $E^+$ and $E^-$ that contribute to relevant solutions in the domain?
Acknowledgements

- Sebastian Uchitel, Imperial College London
- Jeff Kramer, Imperial College London
- Alessandra Russo, Imperial College London
- Axel van Lamsweerde, UCL, Belgium


Thank You