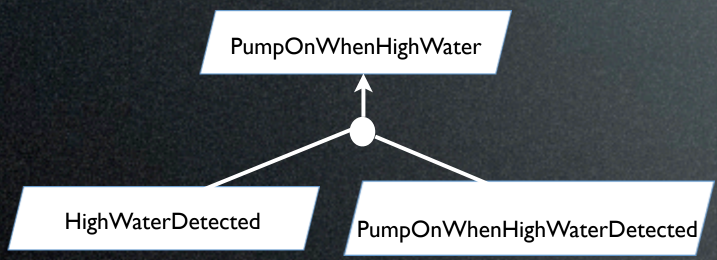
Requirements Elaboration: An Inductive Search Problem

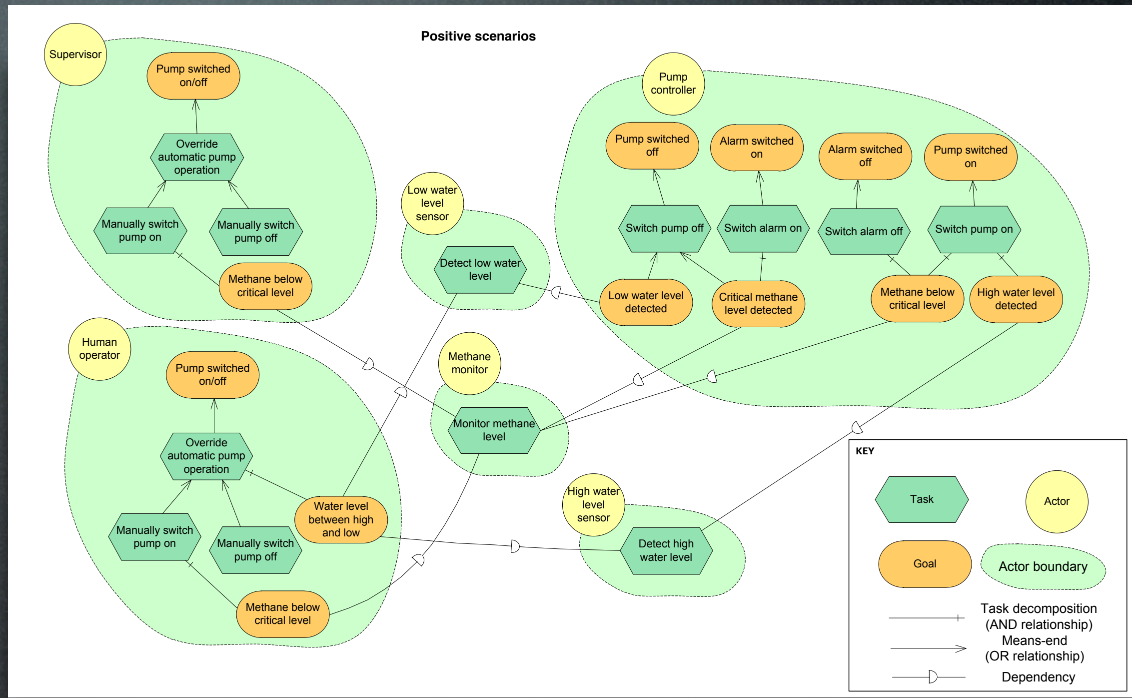
Dalal Alrajeh
Department of Computing
Imperial College London

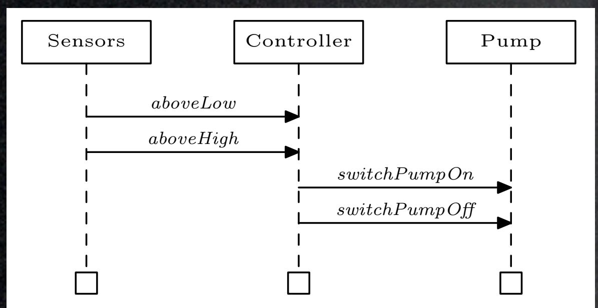
The 25th CREST Open Workshop: Requirements and Test Optimisation

February 12th, 2013

Requirements







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)

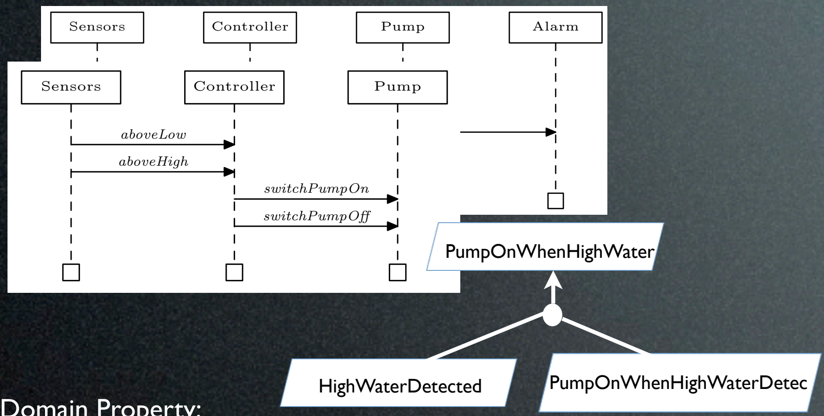
- Scenarios

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

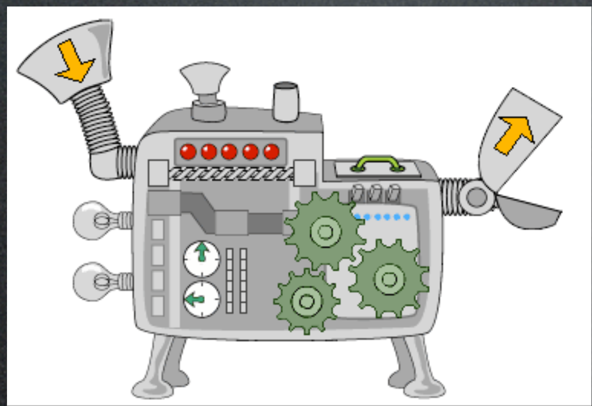
- Operational Requirements

  Constraints on the actions an agent can perform to achieve the

Requirements to Operational Requirements



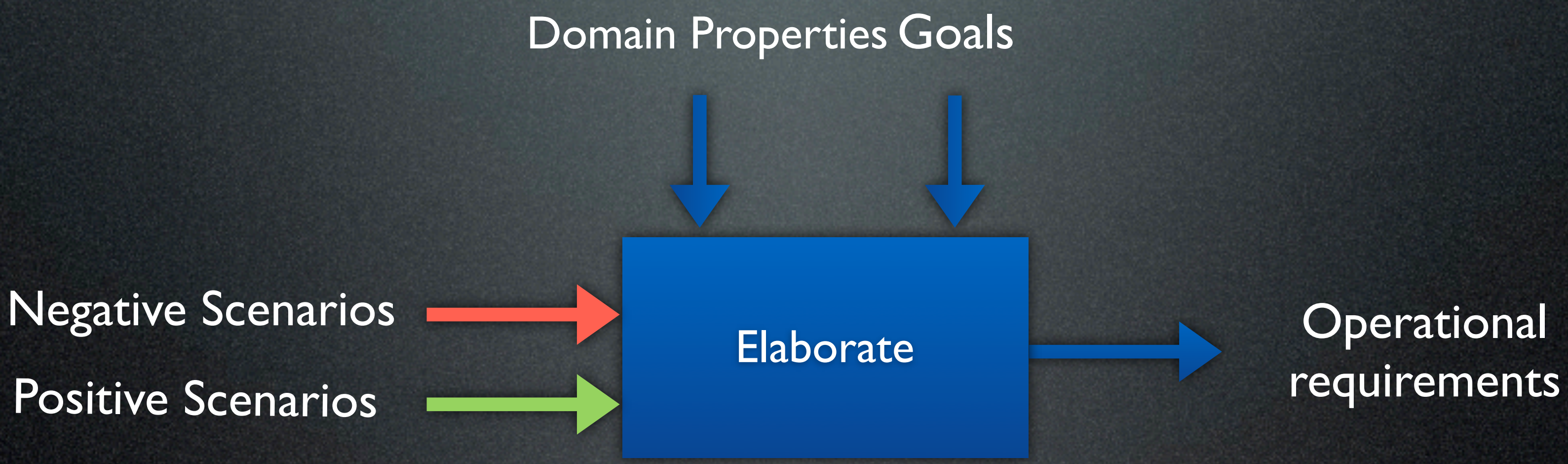
Domain Property:
Operation: switchPumpOn
DomPre: ¬PumpOn
DomPost: PumpOn



Operational requirements on software agents

$\square(\text{CriticalMethane} \rightarrow \bigcirc \neg \text{switchPumpOn})$
$\square((\text{CriticalMethane} \wedge \text{PumpOn}) \rightarrow \bigcirc \text{switchPumpOff})$
$\square(\text{HighWater} \wedge \neg \text{CriticalMethane} \wedge \neg \text{PumpOn} \rightarrow \bigcirc \text{switchPumpOn})$
$\square((\text{HighWater} \wedge \neg \text{CriticalMethane}) \rightarrow \bigcirc \neg \text{switchPumpOff})$
$\square(\text{LowWater} \rightarrow \bigcirc \neg \text{switchPumpOn})$
$\square((\text{LowWater} \wedge \text{PumpOn}) \rightarrow \bigcirc \neg \text{switchPumpOff})$
$\square((\text{CriticalMethane} \wedge \neg \text{Alarm}) \rightarrow \bigcirc \text{raiseAlarm})$
$\square((\neg \text{CriticalMethane} \wedge \text{Alarm}) \rightarrow \bigcirc \text{stopAlarm})$

Operational Requirements Elaboration



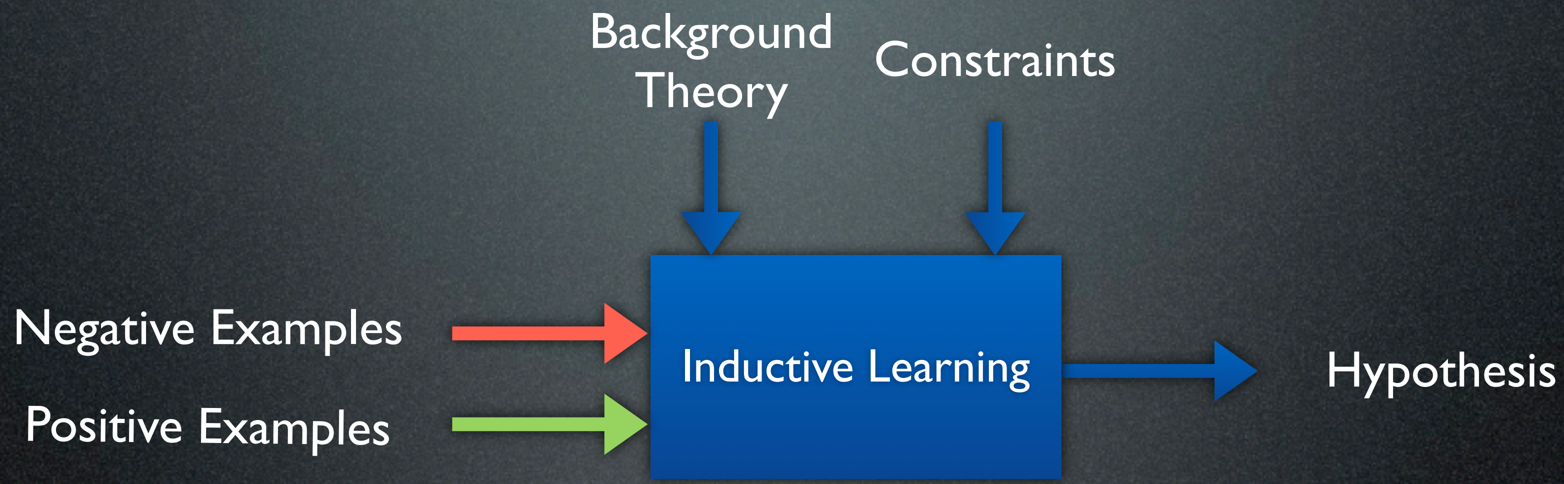
Automated

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:

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

Find:

$$H1 = \{fly(X) \leftarrow bird(X).\}$$

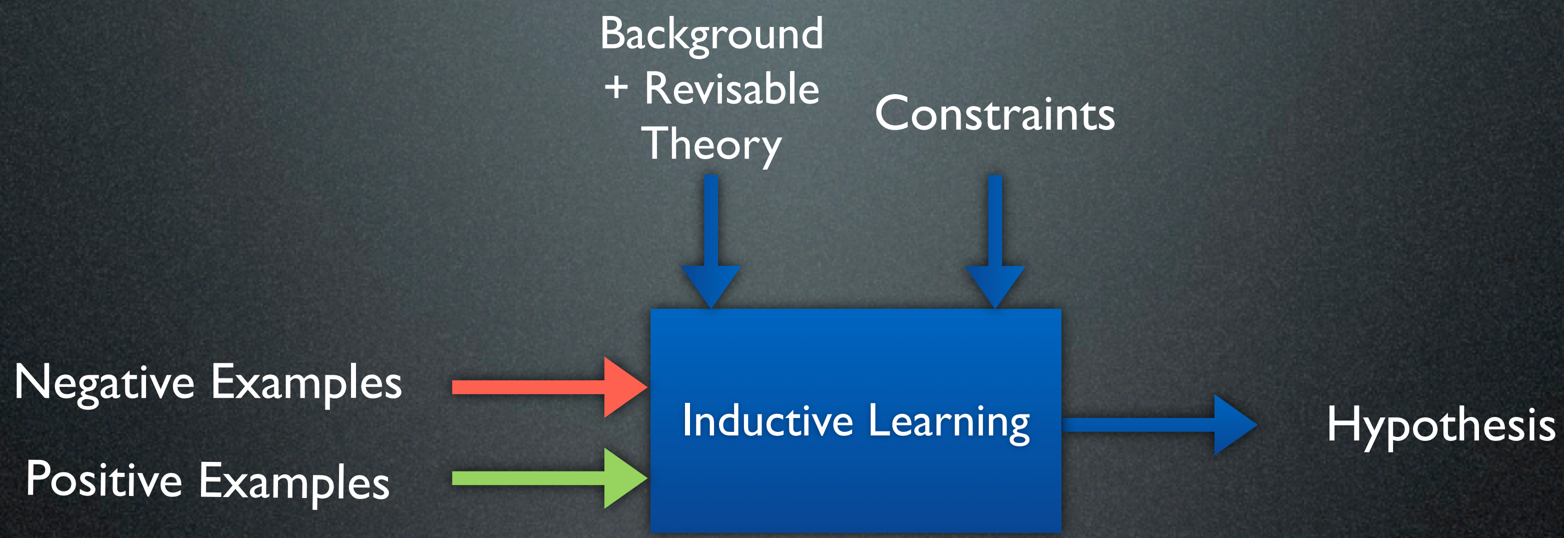
$$H2 = \{fly(X) \leftarrow animal(X).\}$$

$$H3 = \{fly(X).\}$$

Search space includes rules with the predicate *fly* in the head and the predicates *bird* and *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

revised
The secret of flight

Given:

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

Find:

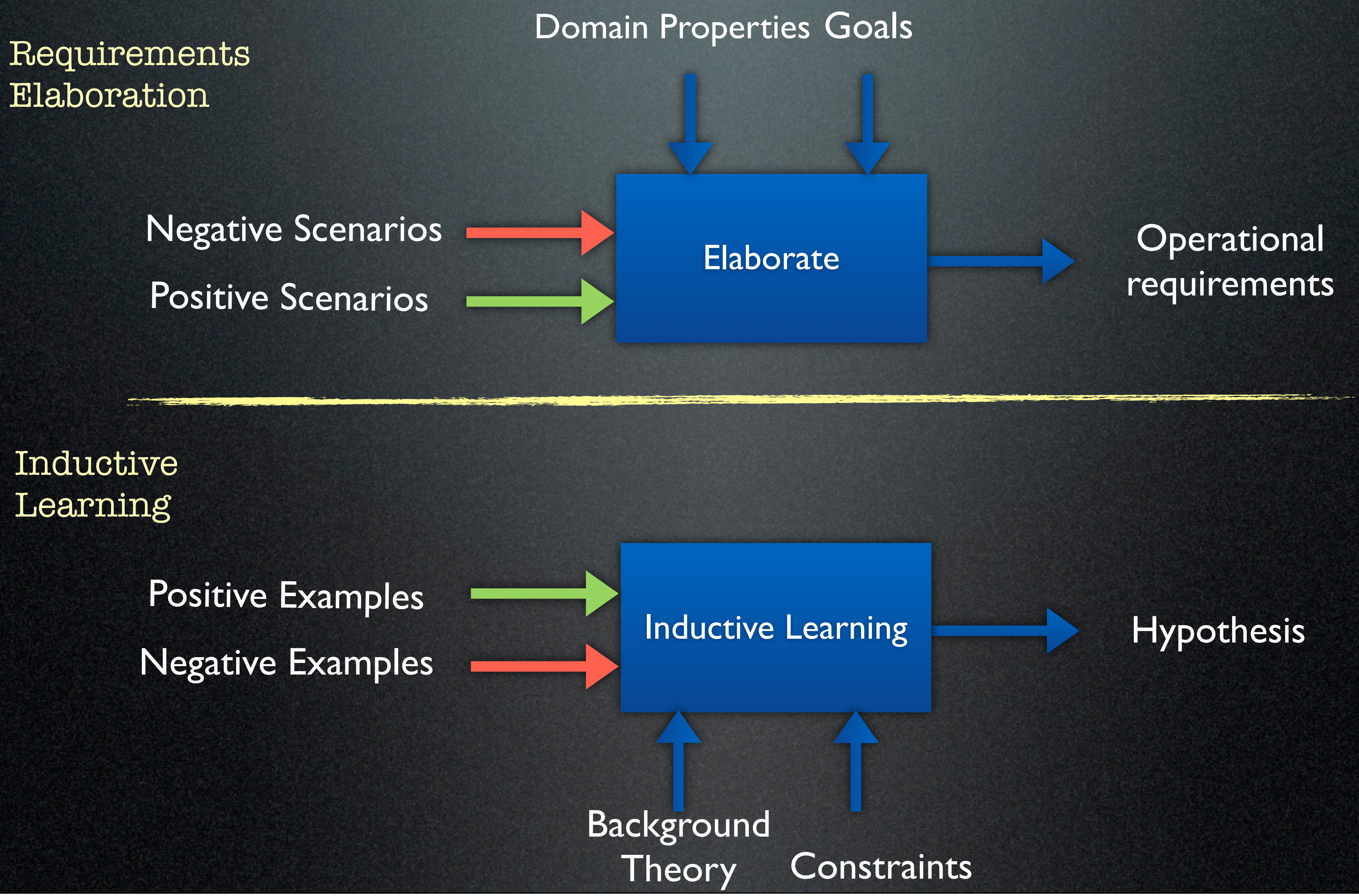
$H_1 = \{fly(X) \leftarrow not\ penguin(X).\}$

$c(R,R`) = 1$

Search space includes rules with the predicate *fly* in the head and the predicates *bird* and *penguin* in the body.

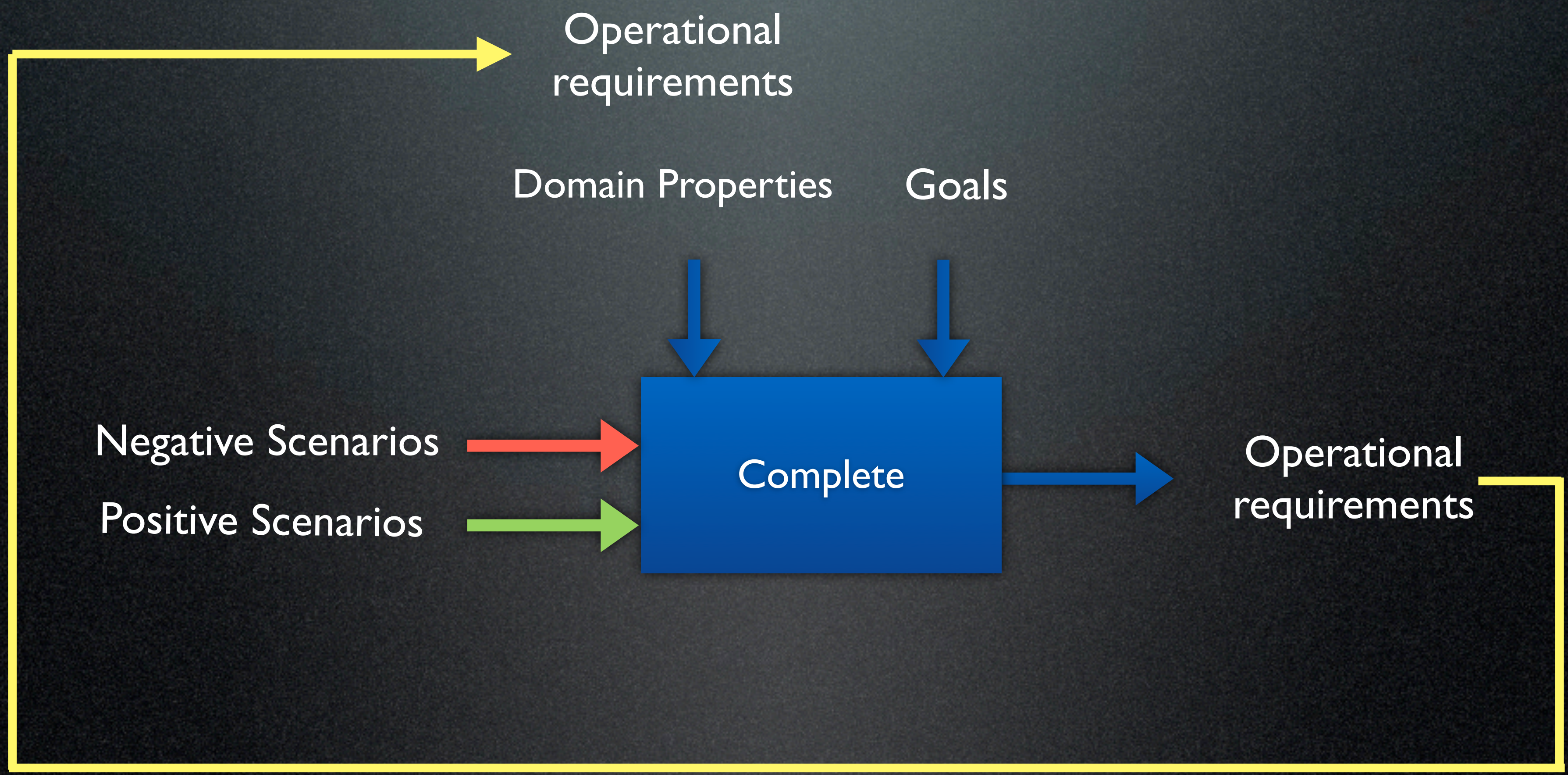
Optimal Solution is the hypothesis with minimal changes min(c(R,R`))

Elaboration & Learning: Symmetrical?



Learning Operational Requirements

Dalal Alrajeh, Jeff Kramer, Alessandra Russo, Sebastian Uchitel. *Learning Operational Requirements from Goal Models*, Proceedings of 31st International Conference on Software Engineering (ICSE'09), 265-275, 2009.



Learning Task

Given

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

$$D \cup O \not\models s^+ \quad \text{for some } s^+ \text{ in } S^+$$
$$D \cup O \models s^- \quad \text{for some } s^- \text{ in } S^-$$
$$D \cup O \cup G \quad \text{is consistent}$$

Find

The smallest set of operational requirements O\` such that

$$D \cup O \cup O` \models s^+ \quad \text{for all } s^+ \text{ in } S^+$$
$$D \cup O \cup O` \not\models s^- \quad \text{for all } s^- \text{ in } S^-$$
$$D \cup O \cup O` \cup G \quad \text{is consistent}$$

where $\models$ is interpreted as the linear temporal logic satisfaction relation with respect to traces in the semantic model (i.e. an LTS of ($D \cup O \cup O`$)).

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 pre-set high water level and deactivated once it reaches low water level.

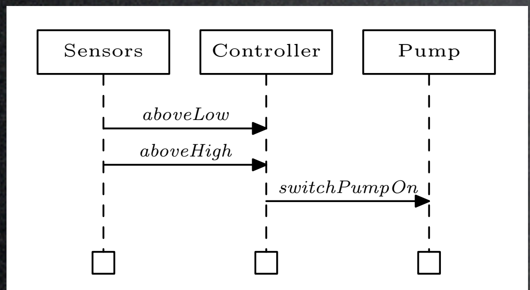
Moreover, the pump must be switched off if the percentage of methane in the mine exceeds a certain critical limit.

Mine Pump Example

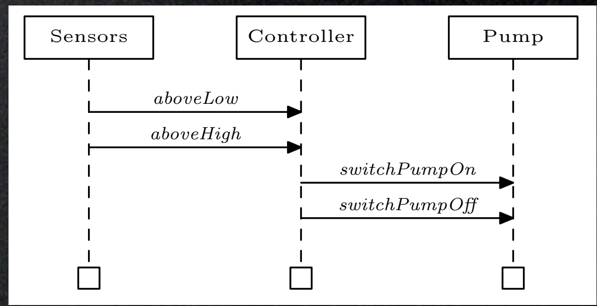
Goal: Achieve[PumpOnWhenHighWaterAndNoMethane]
Informal Definition: The pump shall be on when the water level is above high water level and there is no methane present in the mine
Formal Definition
(SYN) $\square((HighWater \wedge \neg CriticalMethane) \rightarrow \bigcirc PumpOn)$

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

Positive Example



Negative Example

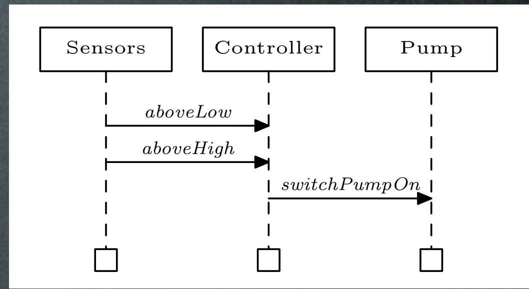


Coverage Check

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

Operational Requirement:

$\models$

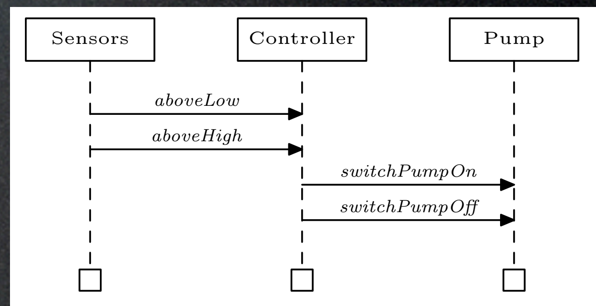


?

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

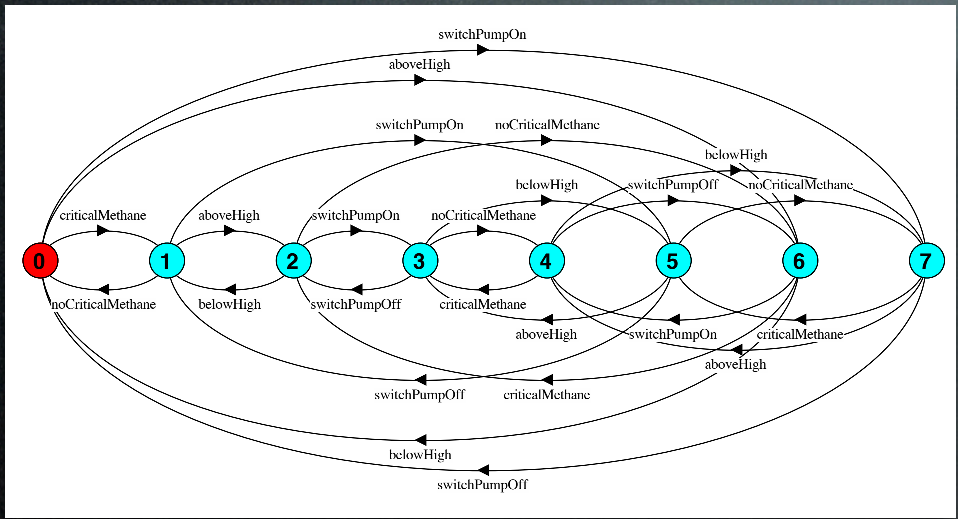
Operational Requirement:

$\not\models$

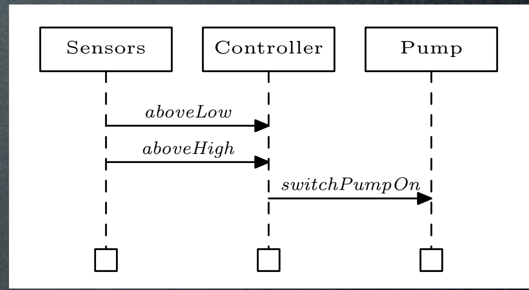


?

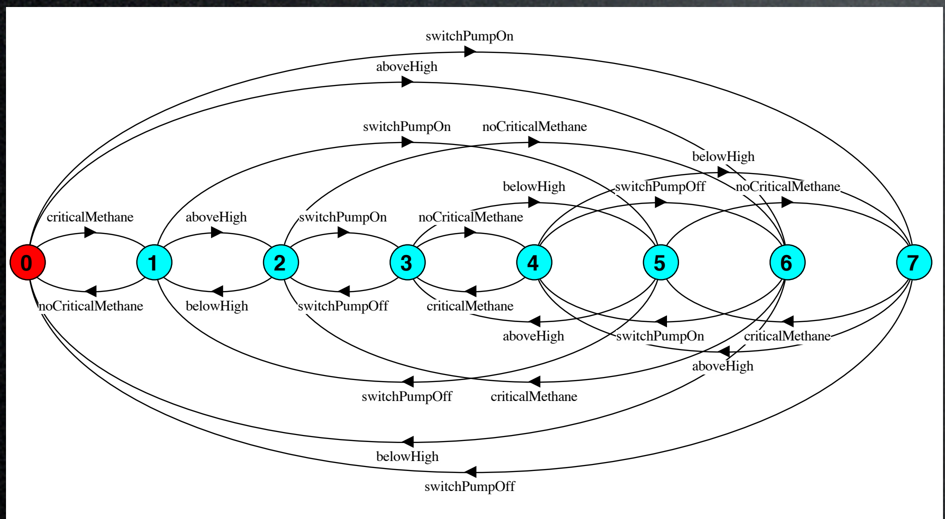
Coverage Check



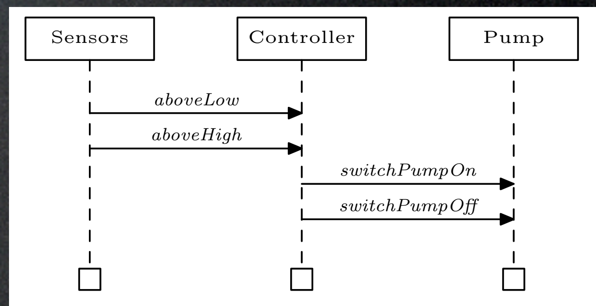
$\models$



?



$\not\models$

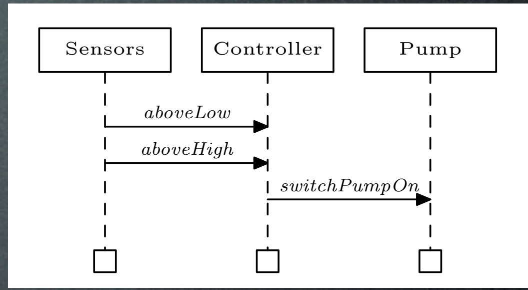


?

Learned Requirements

Domain Property:
Operation: switchPumpOn
DomPre: $\neg$PumpOn DomPost: PumpOn
Operation: switchPumpOff
DomPre: PumpOn DomPost: $\neg$PumpOn
Operation: aboveHigh
DomPre: $\neg$ HighWater DomPost: HighWater

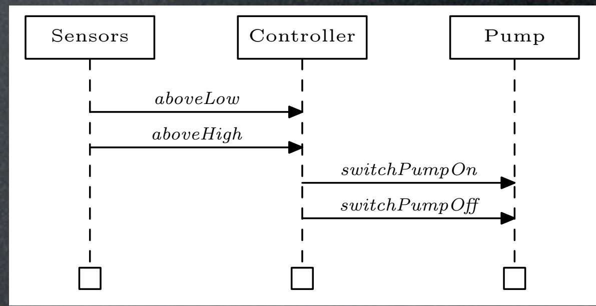
$\models$



Operational Requirement:

Operation: switchPumpOff

$\not\models$



H_1 = ReqPre: HighWater $\wedge$ $\neg$CriticalMethane
$\Box$((HighWater $\wedge$ $\neg$CriticalMethane) $\rightarrow$ $\bigcirc$ $\neg$ switchPumpOff)

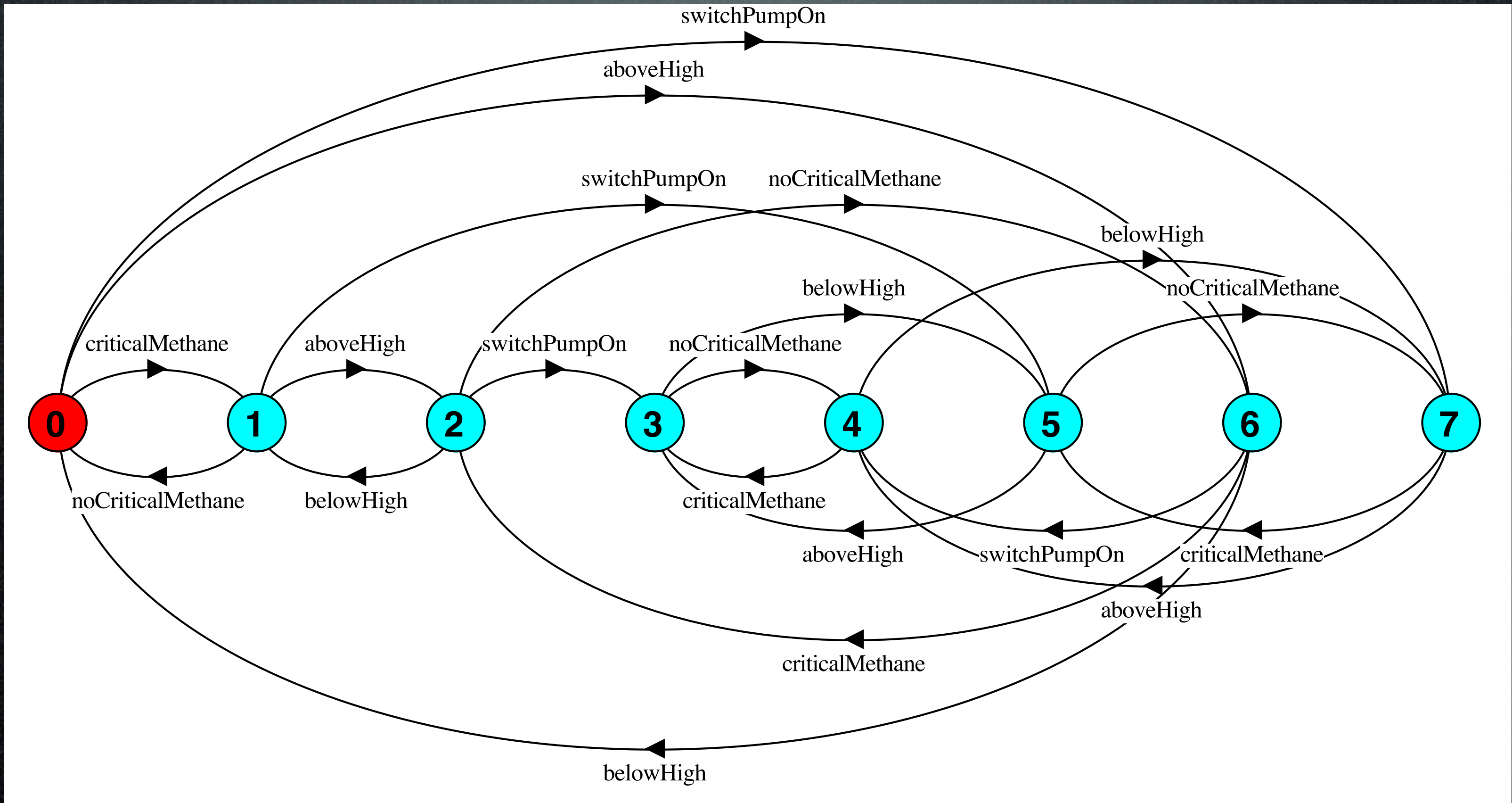
H_2 = ReqPre: HighWater
$\Box$((HighWater) $\rightarrow$ $\bigcirc$ $\neg$ switchPumpOff)

H_3 = ReqPre: $\neg$CriticalMethane
$\Box$(($\neg$CriticalMethane) $\rightarrow$ $\bigcirc$ $\neg$ switchPumpOff)

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

Learning method returns minimal solution that would prevent s^- from occurring

Minimal == Optimal ?

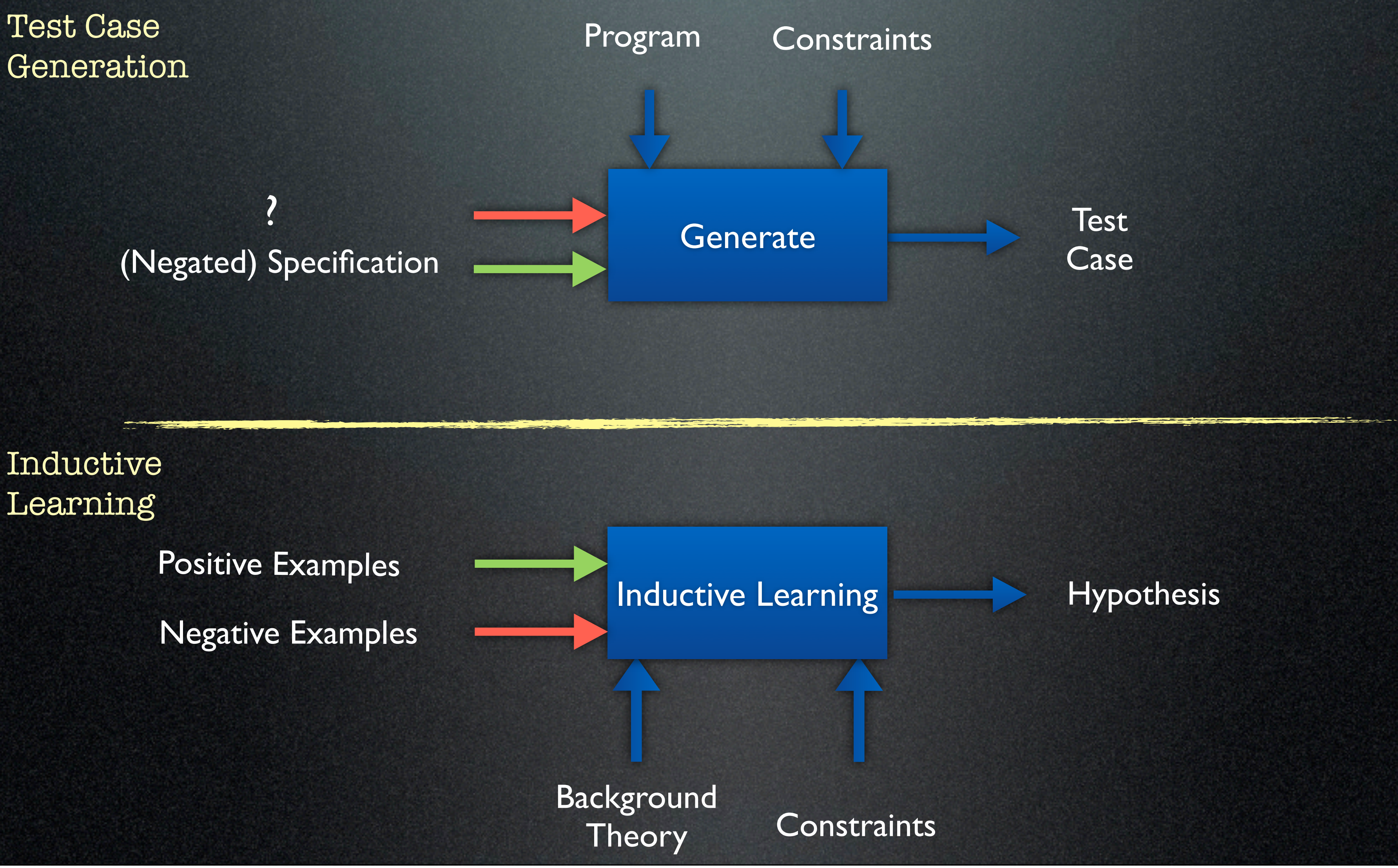


Never switch the pump off

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

Given

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

P U S U C is consistent

Find

The smallest Test Case T such that

P U T $\not\models$ s for some s in S

P U S U C U 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

References

Dalal Alrajeh, Jeff Kramer, Alessandra Russo and Sebastian Uchitel. Elaborating Requirements using Model Checking and Inductive Learning. *In IEEE Transactions on Software Engineering, 2012.*

Dalal Alrajeh, Jeff Kramer, Axel van Lamsweerde, Alessandra Russo and Sebastian Uchitel. *Generating Obstacle Conditions for Requirements Completeness*. Proceedings of 34th International Conference on Software Engineering (ICSE'12), 705-0715, 2012.

Dalal Alrajeh, Jeff Kramer, Alessandra Russo and Sebastian Uchitel *Learning from Vacuously Satisfiable Scenario-based Specifications*. Proceedings of 15th International Conference on Fundamental Approaches to Software Engineering (FASE'12), 377-393, 2012.

Dalal Alrajeh, Jeff Kramer, Alessandra Russo and Sebastian Uchitel *An Inductive Approach for Modal Transition Systems Refinement*, Technical Communications of the 27th International Conference on Logic Programming (ICLP'11), 106-116, 2011.

Dalal Alrajeh, Jeff Kramer, Alessandra Russo, Sebastian Uchitel. *Learning Operational Requirements from Goal Models*, Proceedings of 31st International Conference on Software Engineering (ICSE'09), 265-275, 2009.

Thank You