Automated Goal Operationalisation based on Interpolation and SAT Solving

Renzo Degiovanni, Dalal Alrajeh, Nazareno Aguirre and Sebastian Uchitel

{rdegiovanni, naguirre}@dc.exa.unrc.edu.ar
{dalal.alrajeh04, s.uchitel}@imperial.ac.uk

Automated Goal Operationalisation based on Interpolation and SAT Solving

Renzo Degiovanni, Dalal Alrajeh, Nazareno Aguirre and Sebastian Uchitel

{rdegiovanni, naguirre}@dc.exa.unrc.edu.ar
{dalal.alrajeh04, s.uchitel}@imperial.ac.uk

Automated Goal Operationalisation based on Interpolation and SAT Solving

Renzo Degiovanni, Dalal Alrajeh, Nazareno Aguirre and Sebastian Uchitel

{rdegiovanni, naguirre}@dc.exa.unrc.edu.ar
{dalal.alrajeh04, s.uchitel}@imperial.ac.uk

Automated Goal Operationalisation based on Interpolation and SAT Solving

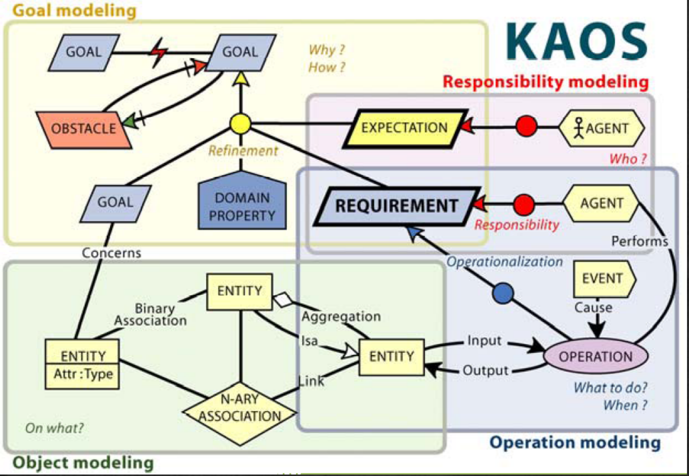
Renzo Degiovanni, Dalal Alrajeh, Nazareno Aguirre and Sebastian Uchitel

{rdegiovanni, naguirre}@dc.exa.unrc.edu.ar
{dalal.alrajeh04, s.uchitel}@imperial.ac.uk

Goal Oriented Requirements Engineering

- Goals are objectives the system under consideration must achieve.
- GORE refers to the use of goals for requirements elicitation, elaboration, organization, specification, analysis, documentation and evolution.
- KAOS Knowledge Acquisition in autOmated Specifications:
  - a conceptual model for acquiring and structuring requirements models;
  - a set of strategies for elaborating requirements using this framework;
  - an automated assistant to provide guidance in the acquisition process.

KAOS specifications



Goal Model

Objectives the system should meet are defined in this model and interrelated through AND/OR refinement links.

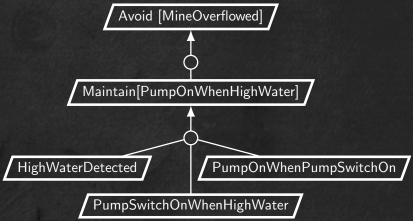


Figura: Goal Model.

Goal Model

Objectives the system should meet are defined in this model and interrelated through AND/OR refinement links.

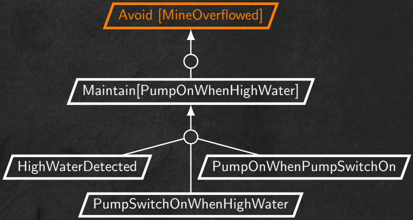


Figura: Goal Model.

- High level goals

Goal Model

Objectives the system should meet are defined in this model and interrelated through AND/OR refinement links.

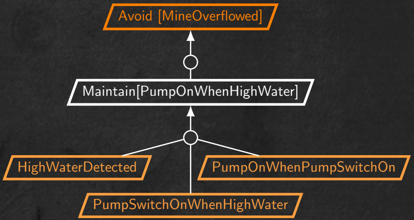


Figura: Goal Model.

- High level goals
- Low level goals

Goal Model

Objectives the system should meet are defined in this model and interrelated through AND/OR refinement links.

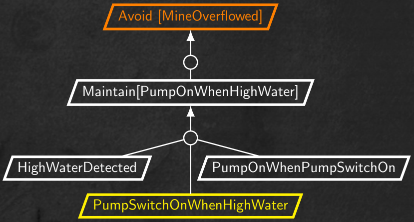


Figura: Goal Model.

- High level goals
- Low level goals
- Requirements

Goal Model

Objectives the system should meet are defined in this model and interrelated through AND/OR refinement links.

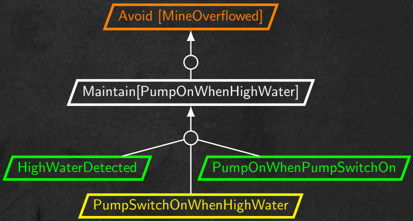


Figura: Goal Model.

- High level goals
- Low level goals
- Requirements
- Expectations

Goal Model

Objectives the system should meet are defined in this model and interrelated through AND/OR refinement links.

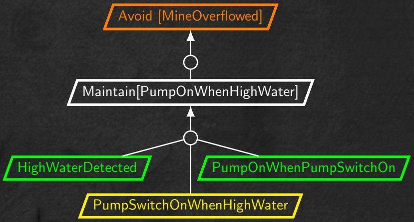


Figura: Goal Model.

- High level goals
- Low level goals
- Requirements
- Expectations
- Goal patterns:
  - Avoid: $C \Rightarrow \neg T$
  - Maintain: $C \Rightarrow T$

Object Model

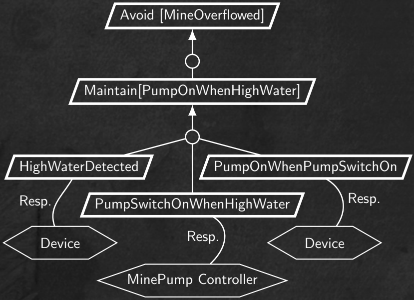
This model defines the domain entities, relationships and attributes that are relevant to goal formulations.

Figura: Object Model.

- Objects can be specified formally by means of domain invariants.

Responsibility Model

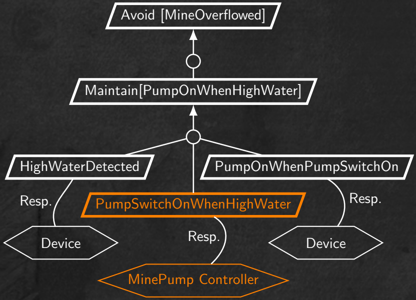
The agents in the system are described together with their interfaces and responsibilities with respect to the goals.



- Responsibility assignments provide a criterion for stopping the goal refinement process.
- A goal assigned as the responsibility of a single agent must not be refined further.

Responsibility Model

The agents in the system are described together with their interfaces and responsibilities with respect to the goals.



- Responsibility assignments provide a criterion for stopping the goal refinement process.
- A goal assigned as the responsibility of a single agent must not be refined further.

Operation Model

This model defines the services to be provided by software agents.

```
Operation switchPumpOn
   DomPre   !PumpOn
   DomPost  PumpOn
   ReqPre   True
   ReqTrig  False
```

- input-output relations over components of the object model.
- operation applications define state transitions.

Operation Model

This model defines the services to be provided by software agents.

```
Operation switchPumpOn
   DomPre   !PumpOn
   DomPost  PumpOn
   ReqPre   True
   ReqTrig  False
```

- input-output relations over components of the object model.
- operation applications define state transitions.
- domain conditions capture the elementary state transitions in the domain.

Operation Model

This model defines the services to be provided by software agents.

```
Operation switchPumpOn
    DomPre   !PumpOn
    DomPost  PumpOn
    ReqPre   True
    ReqTrig  False
```

- input-output relations over components of the object model.
- operation applications define state transitions.
- domain conditions capture the elementary state transitions in the domain.
- required conditions capture additional strengthenings to meet the goals.
  - ReqPre: enabling condition.
  - ReqTrig: triggering condition.

Goal Operationalisation

Operation *switchPumpOn*
  DomPre `!PumpOn`
  DomPost `PumpOn`
  ReqPre `True`
  ReqTrig `False`

Operationalization refers to the process of prescribing additional pre-, trigger-, and postconditions on operations in order to achieve goal specifications.

Safety Goal [PumpOnWhenHighWaterAndNoMethane]

$$\Box(\mathtt{HighWater} \wedge \neg\mathtt{Methane} \rightarrow \bigcirc\mathtt{PumpOn})$$

Goal Operationalisation

Operation *switchPumpOn*
DomPre !PumpOn
DomPost PumpOn
ReqPre True
ReqTrig False

A goal is correctly operationalised by a set of operations, if satisfying all required conditions in the set guarantees the satisfaction of the goal.

Safety Goal [PumpOnWhenHighWaterAndNoMethane]

$$\square(\mathtt{HighWater} \wedge \neg \mathtt{Methane} \rightarrow \bigcirc \mathtt{PumpOn})$$

Goal Operationalisation

Operation *switchPumpOn*
DomPre !PumpOn
DomPost PumpOn
ReqPre True
ReqTrig
HighWater & !Methane

A goal is correctly operationalised by a set of operations, if satisfying all required conditions in the set guarantees the satisfaction of the goal.

Safety Goal [PumpOnWhenHighWaterAndNoMethane]

$$\Box(\texttt{HighWater} \wedge \neg\texttt{Methane} \rightarrow \bigcirc\texttt{PumpOn})$$

Interpolation

Given two sets A and B of formulas such that $A \cup B$ is unsatisfiable, an interpolant for A and B is a formula I such that:

- I is `true` in all models of A,
- I is `false` in all models of B, and
- I is in $\mathcal{L}(A) \cap \mathcal{L}(B)$, i.e., the common language of A and B.

Simple Example

$A \equiv p \wedge q \qquad B \equiv \neg q \wedge r \qquad I \equiv q$

Interpolation to Refine Requirements

Suppose we obtain a counterexample trace T, violating a particular goal G.

- If we build a formula F_T capturing trace T, and a formula F_G capturing the fact that G holds at the last state
    - clearly $F_T \wedge F_G$ is unsatisfiable.
- We can produce an interpolant I from these formulas, that provides a condition whose validity leads to the goal violation.

Interpolation to Refine Requirements

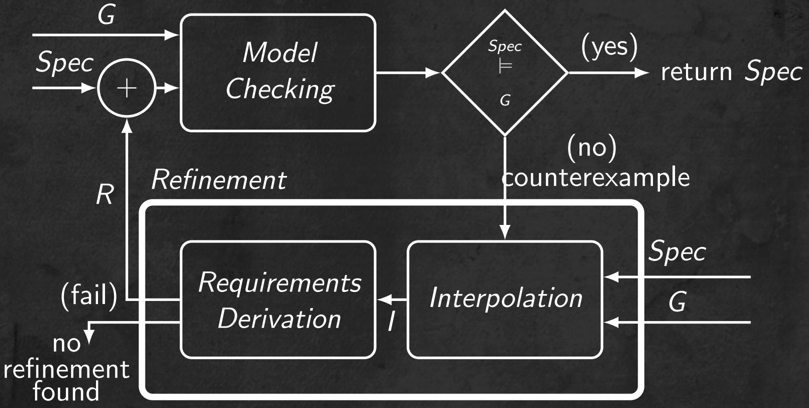
- I is a property of the trace T, which implies the negation of the goal G.
- The interpolant allows us to obtain a weaker "counterexample" than T, a condition reachable from the initial state which leads to the violation of a goal.
  - solely by removing I, we do not guarantee the satisfaction of G,
  - but not removing it guarantees its violation.
- Notice that interpolation is, in some sense, a form of generalisation.

Brief Overview

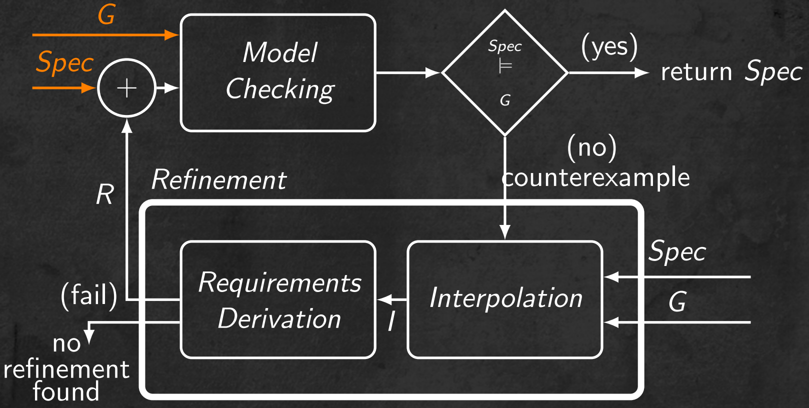
We present an approach for goal operationalisation, that automatically computes required pre/triggering conditions for operations, in order to fulfil a set of goals.

- Iterative,
- base on Interpolation and SAT solving,
- safety and the time progress goals,
- a wide range of liveness goals, namely, reactivity properties.

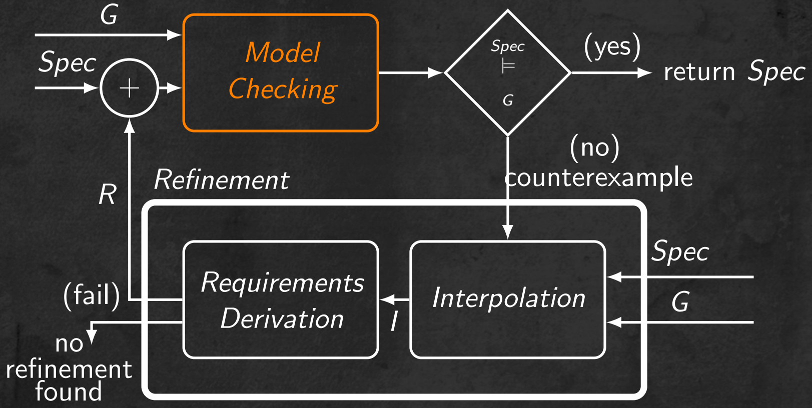
The Approach



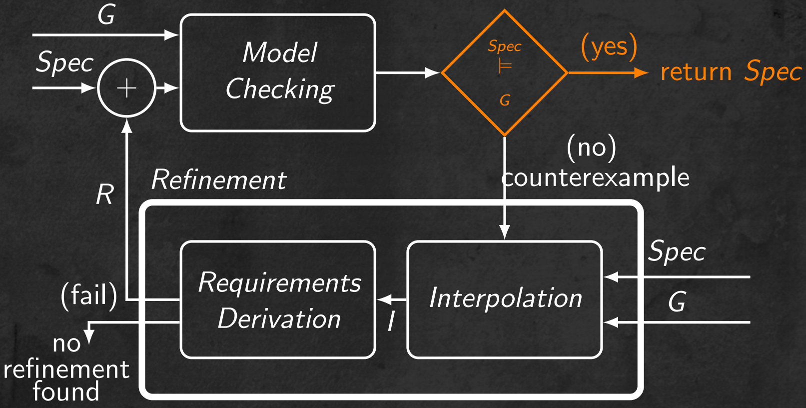
The Approach



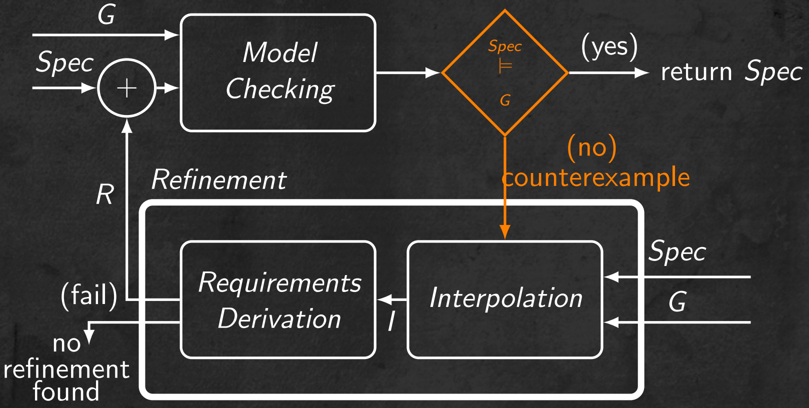
The Approach



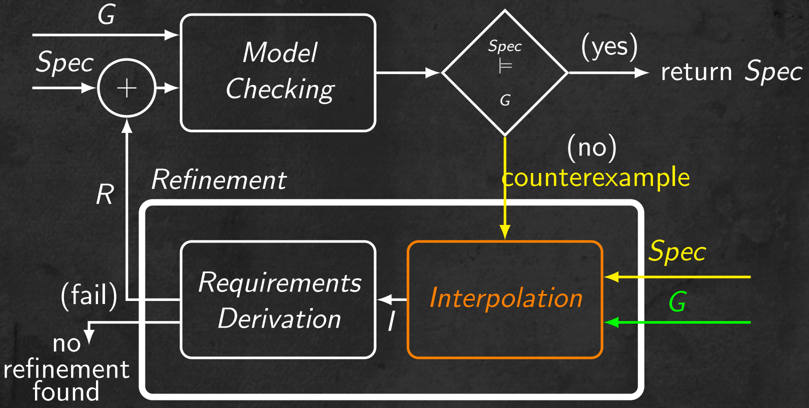
The Approach



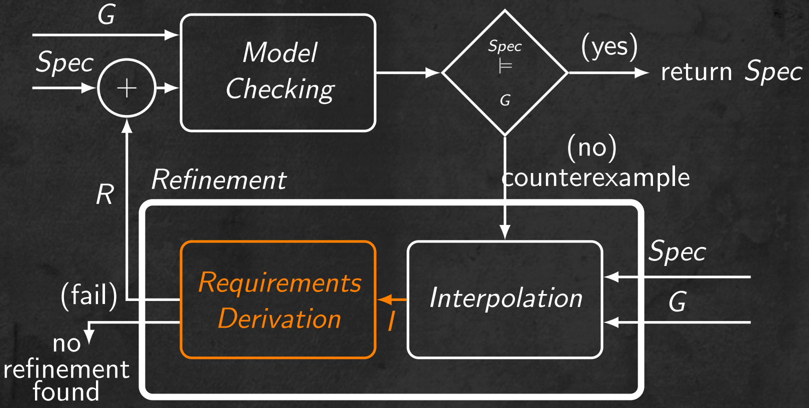
The Approach



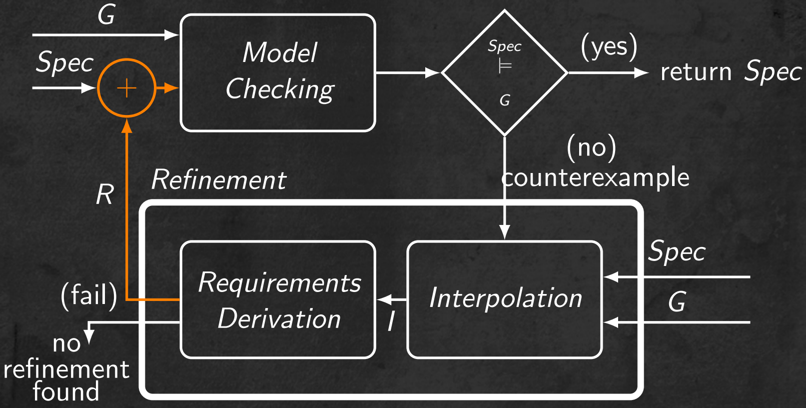
The Approach



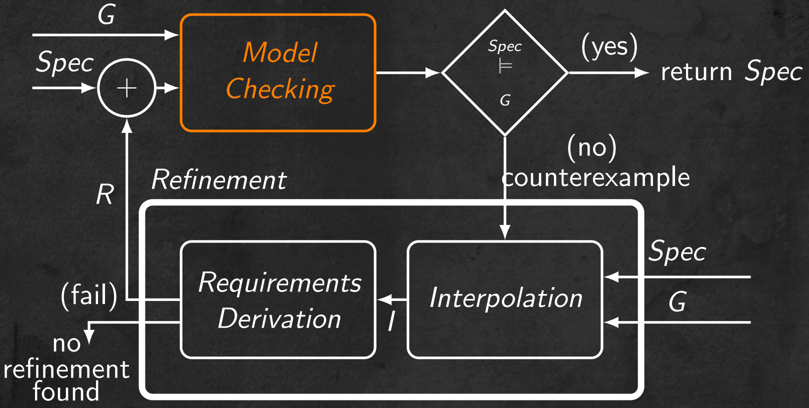
The Approach



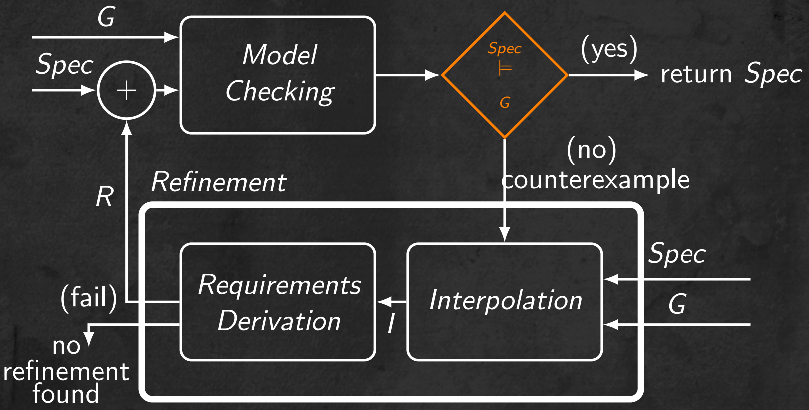
The Approach



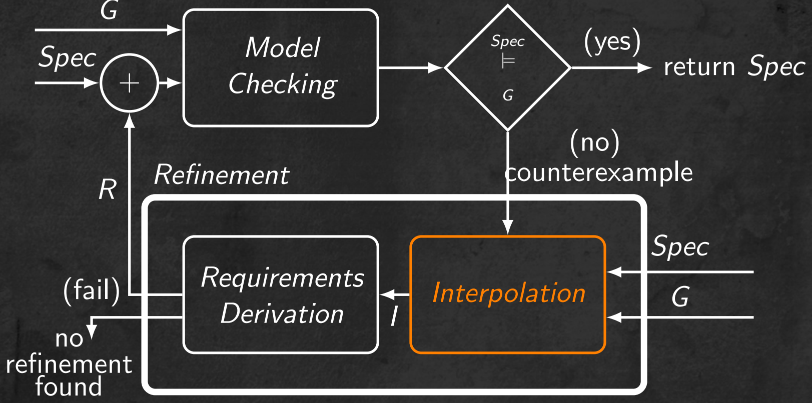
The Approach



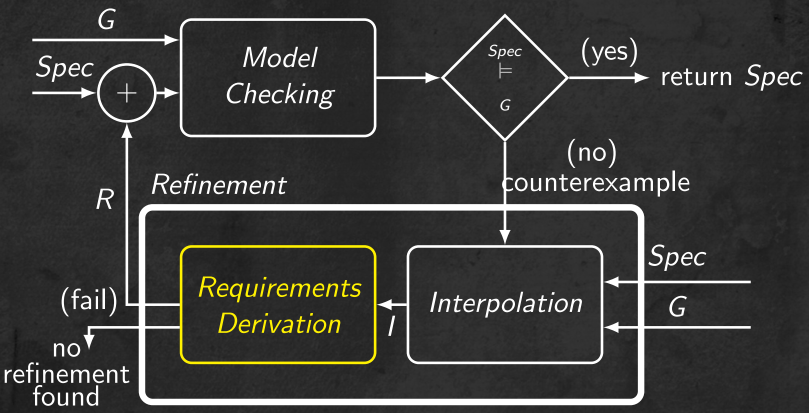
The Approach



The Approach



The Approach

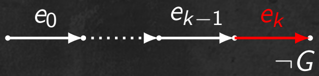


Requirements Derivation Phase

$$\bullet \xrightarrow{e_0} \bullet \cdots\cdots \blacktriangleright \bullet \xrightarrow{e_{k-1}} \bullet \xrightarrow{e_k} \underset{\neg G}{\bullet}$$

A counterexample may be removed either by:

Requirements Derivation Phase



A counterexample may be removed either by:

- prohibiting the occurrence of an operation from certain states
  - i.e., strengthening the operation's required precondition,

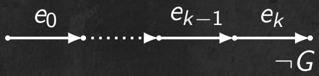
Requirements Derivation Phase

$$\bullet \xrightarrow{e_0} \bullet \cdots\cdots \bullet \xrightarrow{e_{k-1}} \bullet \qquad \bullet$$

A counterexample may be removed either by:

- prohibiting the occurrence of an operation from certain states
  - i.e., strengthening the operation's required precondition,

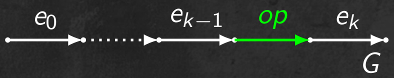
Requirements Derivation Phase



A counterexample may be removed either by:

- prohibiting the occurrence of an operation from certain states
    - i.e., strengthening the operation's required precondition,
- or forcing an operation to occur in certain states
    - i.e., weakening its required triggering condition.

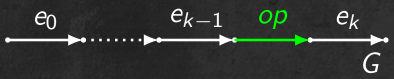
Requirements Derivation Phase



A counterexample may be removed either by:

- prohibiting the occurrence of an operation from certain states
    - i.e., strengthening the operation's required precondition,
- or forcing an operation to occur in certain states
    - i.e., weakening its required triggering condition.

Requirements Derivation Phase



A counterexample may be removed either by:

- prohibiting the occurrence of an operation from certain states
  - i.e., strengthening the operation's required precondition,
- or forcing an operation to occur in certain states
  - i.e., weakening its required triggering condition.

The approach is concerned with automatically detecting which of the above cases is necessary.

- *weakest preconditions play an important role.*

Strengthening Required Preconditions

PumpOffWhenLowWater =
$\Box(\texttt{LowWater} \rightarrow \bigcirc\neg\texttt{PumpOn})$

Counterexample:

tick	LowWater
switchPumpOn	LowWater & PumpOn
tick	LowWater & PumpOn

$I = \texttt{LowWater} \wedge \texttt{PumpOn}$

Strengthening Required Preconditions

PumpOffWhenLowWater =
$\Box(\texttt{LowWater} \rightarrow \bigcirc \neg \texttt{PumpOn})$

Counterexample:

tick	LowWater
switchPumpOn	LowWater & PumpOn
tick	LowWater & PumpOn

$I = \texttt{LowWater} \wedge \texttt{PumpOn}$

Strengthening Required Preconditions

```
PumpOffWhenLowWater =
        □(LowWater → ◯¬PumpOn)
```

```
Counterexample:
tick            LowWater
switchPumpOn    LowWater & PumpOn
tick            LowWater & PumpOn
```

$I = \texttt{LowWater} \wedge \texttt{PumpOn}$

- we compute the weakest precondition of the interpolant with respect to the last operation.

Strengthening Required Preconditions

```
PumpOffWhenLowWater =
        □(LowWater → ◯¬PumpOn)
```

```
Counterexample:
tick          LowWater
switchPumpOn  LowWater & PumpOn
tick          LowWater & PumpOn
```

$$I = \texttt{LowWater} \wedge \texttt{PumpOn}$$

$$I' = WP(I, \texttt{switchPumpOn}) = \texttt{LowWater}$$

- we compute the weakest precondition of the interpolant with respect to the last operation.

Strengthening Required Preconditions

PumpOffWhenLowWater =
$\Box$(LowWater $\rightarrow \bigcirc\neg$PumpOn)

Counterexample:

tick	LowWater
switchPumpOn	LowWater & PumpOn
tick	LowWater & PumpOn

$I = \texttt{LowWater} \wedge \texttt{PumpOn}$

$I' = WP(I, \texttt{switchPumpOn}) = \texttt{LowWater}$

- we compute the weakest precondition of the interpolant with respect to the last operation.
- the operation switchPumpOn is able to control the value of the interpolant.

Strengthening Required Preconditions

PumpOffWhenLowWater = $\Box(\texttt{LowWater} \rightarrow \bigcirc\neg\texttt{PumpOn})$

Counterexample:

tick	LowWater
switchPumpOn	LowWater & PumpOn
tick	LowWater & PumpOn

$I = \texttt{LowWater} \wedge \texttt{PumpOn}$

$I' = WP(I, \texttt{switchPumpOn}) = \texttt{LowWater}$

- we compute the weakest precondition of the interpolant with respect to the last operation.
- the operation switchPumpOn is able to control the value of the interpolant.
- then by forbidding switchPumpOn to occur when LowWater, we get rid of this counterexample, and contribute to stop violating G.

Strengthening Required Preconditions

```
PumpOffWhenLowWater =
        □(LowWater → ◯¬PumpOn)
```

```
Counterexample:
tick           LowWater
switchPumpOn   LowWater & PumpOn
tick           LowWater & PumpOn
```

$I = \texttt{LowWater} \wedge \texttt{PumpOn}$

$I' = WP(I, \texttt{switchPumpOn}) = \texttt{LowWater}$

- we compute the weakest precondition of the interpolant with respect to the last operation.
- the operation `switchPumpOn` is able to control the value of the interpolant.
- then by forbidding `switchPumpOn` to occur when `LowWater`, we get rid of this counterexample, and contribute to stop violating G.
  - check it is not obliged to be executed when `LowWater`.

Strengthening Required Preconditions

PumpOffWhenLowWater =
$\Box(\texttt{LowWater} \rightarrow \bigcirc\neg\texttt{PumpOn})$

Counterexample:

tick	LowWater
switchPumpOn	LowWater & PumpOn
tick	LowWater & PumpOn

$I = \texttt{LowWater} \wedge \texttt{PumpOn}$

$I' = WP(I, \texttt{switchPumpOn}) = \texttt{LowWater}$

$ReqPre(\texttt{switchPumpOn}) = \neg\texttt{LowWater}$

- we compute the weakest precondition of the interpolant with respect to the last operation.
- the operation switchPumpOn is able to control the value of the interpolant.
- then by forbidding switchPumpOn to occur when LowWater, we get rid of this counterexample, and contribute to stop violating G.
  - check it is not obliged to be executed when LowWater.

Weakening Required Triggering Conditions

PumpOnWhenHighWaterAndNoMethane =

$\square(\mathtt{HighWater} \land \neg\mathtt{Methane} \rightarrow \bigcirc\mathtt{PumpOn})$

```
Counterexample:
tick    aboveLow    tick
aboveHigh  HighWater
tick       HighWater
tick       HighWater
```

$I = \mathtt{HighWater} \land \neg\mathtt{Methane} \land \neg\mathtt{PumpOn}$

Weakening Required Triggering Conditions

PumpOnWhenHighWaterAndNoMethane =

$\Box(\mathtt{HighWater} \wedge \neg\mathtt{Methane} \rightarrow \bigcirc\mathtt{PumpOn})$

```
Counterexample:
tick     aboveLow    tick
aboveHigh  HighWater
tick       HighWater
tick       HighWater
```

$I = \mathtt{HighWater} \wedge \neg\mathtt{Methane} \wedge \neg\mathtt{PumpOn}$

Weakening Required Triggering Conditions

PumpOnWhenHighWaterAndNoMethane =

$\square(\mathtt{HighWater} \wedge \neg \mathtt{Methane} \rightarrow \bigcirc \mathtt{PumpOn})$

```
Counterexample:
tick      aboveLow    tick
aboveHigh   HighWater
tick        HighWater
tick        HighWater
```

$I = \mathtt{HighWater} \wedge \neg \mathtt{Methane} \wedge \neg \mathtt{PumpOn}$

Weakening Required Triggering Conditions

PumpOnWhenHighWaterAndNoMethane =

$\Box(\texttt{HighWater} \wedge \neg\texttt{Methane} \rightarrow \bigcirc\texttt{PumpOn})$

```
Counterexample:
tick      aboveLow   tick
aboveHigh HighWater
tick      HighWater
tick      HighWater
```

$I = \texttt{HighWater} \wedge \neg\texttt{Methane} \wedge \neg\texttt{PumpOn}$

- No operation executed can control the value of the interpolant.

Weakening Required Triggering Conditions

PumpOnWhenHighWaterAndNoMethane =

$\square(\texttt{HighWater} \wedge \neg\texttt{Methane} \rightarrow \bigcirc\texttt{PumpOn})$

```
Counterexample:
tick     aboveLow   tick
aboveHigh  HighWater
tick       HighWater
tick       HighWater
```

$I = \texttt{HighWater} \wedge \neg\texttt{Methane} \wedge \neg\texttt{PumpOn}$

- No operation executed can control the value of the interpolant.
- We try to remove the counterexample by forcing an operation to occur.

Weakening Required Triggering Conditions

PumpOnWhenHighWaterAndNoMethane =

$\Box(\mathtt{HighWater} \wedge \neg\mathtt{Methane} \rightarrow \bigcirc\mathtt{PumpOn})$

Counterexample:

tick	aboveLow	tick
aboveHigh	HighWater	
tick	HighWater	
tick	HighWater	

$I = \mathtt{HighWater} \wedge \neg\mathtt{Methane} \wedge \neg\mathtt{PumpOn}$

- No operation executed can control the value of the interpolant.
- We try to remove the counterexample by forcing an operation to occur.
- The operation to be triggered, say a_t, must meet two conditions:

Weakening Required Triggering Conditions

PumpOnWhenHighWaterAndNoMethane =

$\Box(\texttt{HighWater} \wedge \neg\texttt{Methane} \rightarrow \bigcirc\texttt{PumpOn})$

Counterexample:

tick	aboveLow	tick
aboveHigh	HighWater	
tick	HighWater	
tick	HighWater	

$I = \texttt{HighWater} \wedge \neg\texttt{Methane} \wedge \neg\texttt{PumpOn}$

- No operation executed can control the value of the interpolant.
- We try to remove the counterexample by forcing an operation to occur.
- The operation to be triggered, say a_t, must meet two conditions:

$$I \Rightarrow DomPre(a_t) \wedge ReqPre(a_t)$$

Weakening Required Triggering Conditions

```
PumpOnWhenHighWaterAndNoMethane =
```

$\square(\texttt{HighWater} \wedge \neg\texttt{Methane} \rightarrow \bigcirc\texttt{PumpOn})$

```
Counterexample:
tick      aboveLow    tick
aboveHigh    HighWater
tick         HighWater
tick         HighWater
```

$I = \texttt{HighWater} \wedge \neg\texttt{Methane} \wedge \neg\texttt{PumpOn}$

- No operation executed can control the value of the interpolant.
- We try to remove the counterexample by forcing an operation to occur.
- The operation to be triggered, say a_t, must meet two conditions:

$$I \Rightarrow DomPre(a_t) \wedge ReqPre(a_t)$$

$$I \wedge \bigcirc DomPost(a_t) \Rightarrow \bigcirc\neg I$$

Weakening Required Triggering Conditions

PumpOnWhenHighWaterAndNoMethane =

$\Box(\texttt{HighWater} \wedge \neg\texttt{Methane} \rightarrow \bigcirc\texttt{PumpOn})$

Counterexample:

```
tick      aboveLow   tick
aboveHigh HighWater
tick      HighWater
tick      HighWater
```

$I = \texttt{HighWater} \wedge \neg\texttt{Methane} \wedge \neg\texttt{PumpOn}$

$ReqTrig(\texttt{switchPumpOn}) =$
$\texttt{HighWater} \wedge \neg\texttt{Methane} \wedge \neg\texttt{PumpOn}$

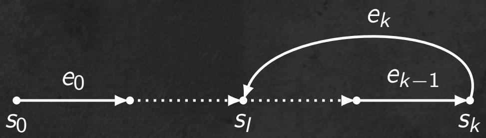
- No operation executed can control the value of the interpolant.
- We try to remove the counterexample by forcing an operation to occur.
- The operation to be triggered, say a_t, must meet two conditions:

$$I \Rightarrow DomPre(a_t) \wedge ReqPre(a_t)$$

$$I \wedge \bigcirc DomPost(a_t) \Rightarrow \bigcirc\neg I$$

Reactivity Liveness Properties

- Manna and Pnueli's characterisation: `([]<>As -> []<>G)`.
- A violation of a property of this kind consists of a prefix (finite part) leading to a loop in which the *As* is satisfied, but not `G`.
- To compute an interpolant for this counterexample, we encode the reactivity goal: $P = (\bigvee_{i=l}^{k} As^i) \Rightarrow (\bigvee_{j=l}^{k} G^j)$



Reactivity Liveness Properties

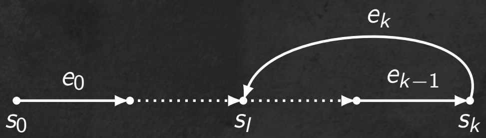
- The interpolant is a weaker representation of the loop part that explains what is wrong in the loop.
- We search for an operation a_t:
    - can be executed at some point in the loop,
    - a_t's execution reaches a states that does not satisfy the interpolant (i.e., "breaks" the loop).

Reactivity Liveness Properties

- If we find a_t, then we refine:

$$ReqTrig(a_t) = ReqTrig^{pre}(a_t) \vee (ReqPre(a_t) \wedge \neg G)$$

- Notice that we do not refine a new triggering condition based on the interpolant.
  - we may produce triggering conditions *weaker* than needed.



Conclusions and Future Works

- We presented an approach for goal operationalisation, that automatically computes required pre-/triggering conditions for operations, in order to fulfil a set of goals.
- The approach is correct, but it is not complete.
- It applies to safety goals and particular kinds of liveness goals, namely reactivity properties.
- We have developed some case studies, and compared with previous approaches.