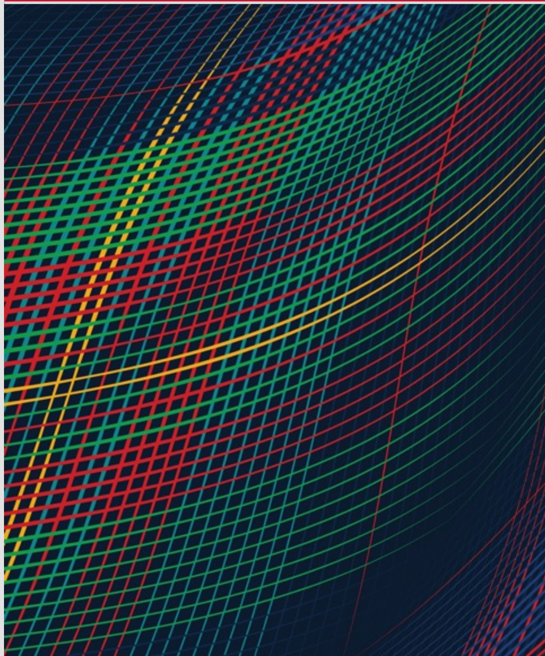
Carnegie Mellon University
Software Engineering Institute

The OSATE Slicer: Graph-Based Reachability for Architectural Models

JULY 20, 2023

Sam Procter



Document Markings

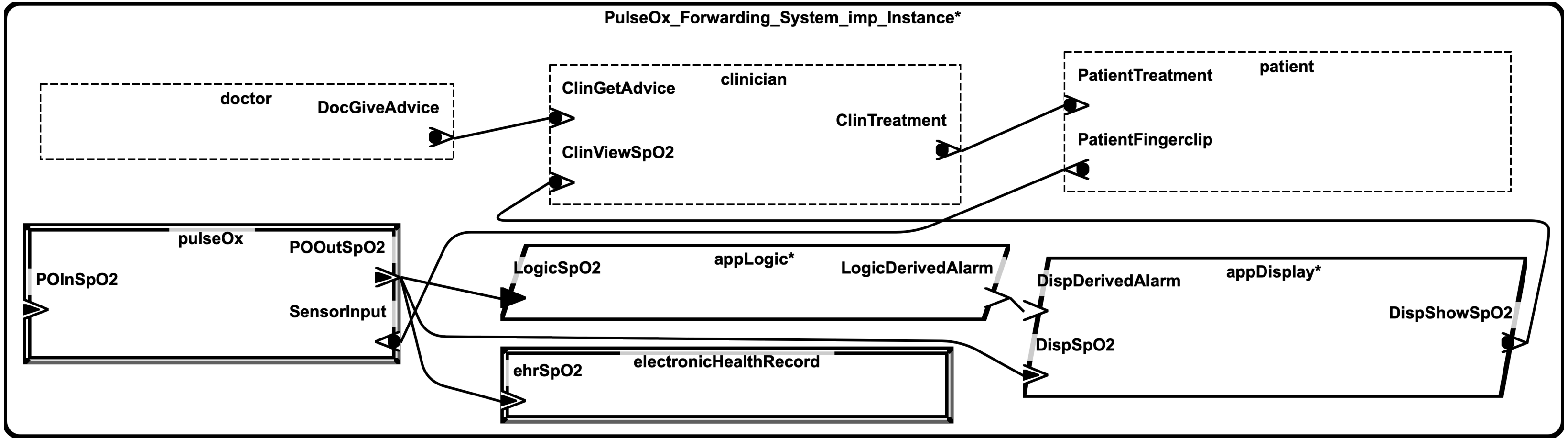
Agenda

- **Introduction & Background**
  - **Problem**
  - **Context**
  - **Solution**
- The OSATE Slicer
- Evaluation

Problem: Models are hard to comprehend

For both manual and automated analyses

- For humans: High cognitive burden – “unwieldy far quicker” than programs [1]
- For automated analyses: Traversal of model elements not easily converted to data- or control-flow ordering

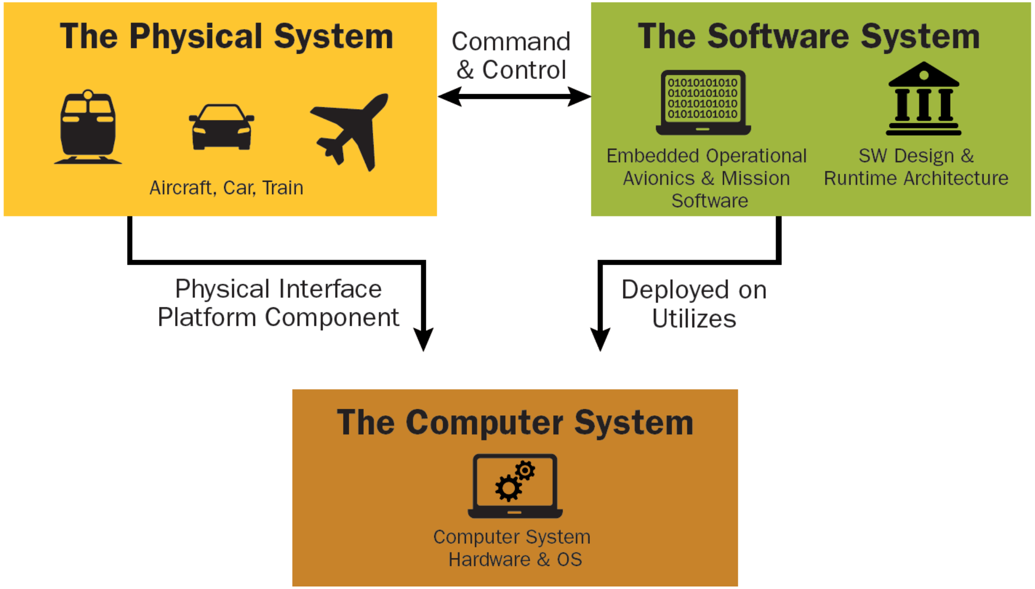


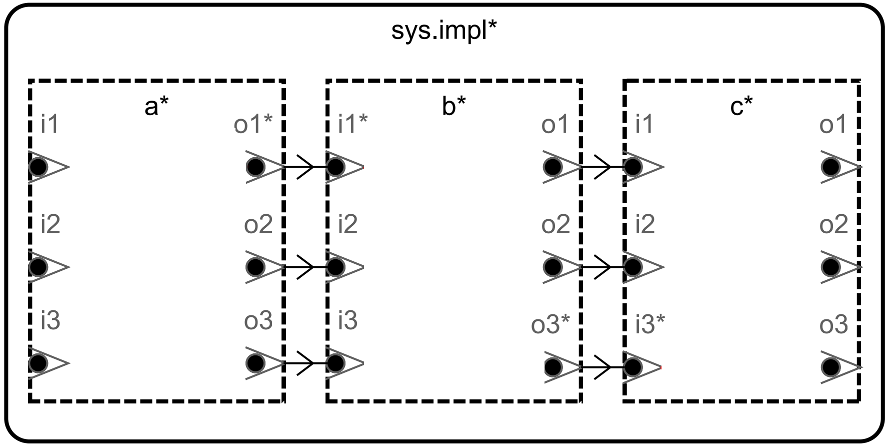
[1] *“State-Based Model Slicing: A Survey.” K. Androutsopoulos, D. Clark, M. Harman, J. Krinke, L. Tratt. ACM Computing Surveys, 2013.*

Context (1): Architecture Analysis & Design Language

AADL

AADL focuses on interaction between the three elements of a software-reliant mission and safety-critical systems


The Physical System
Aircraft, Car, Train
Command & Control
The Software System
01010101010
Embedded Operational Avionics & Mission Software
SW Design & Runtime Architecture
Physical Interface Platform Component
Deployed on Utilizes
The Computer System
Computer System Hardware & OS



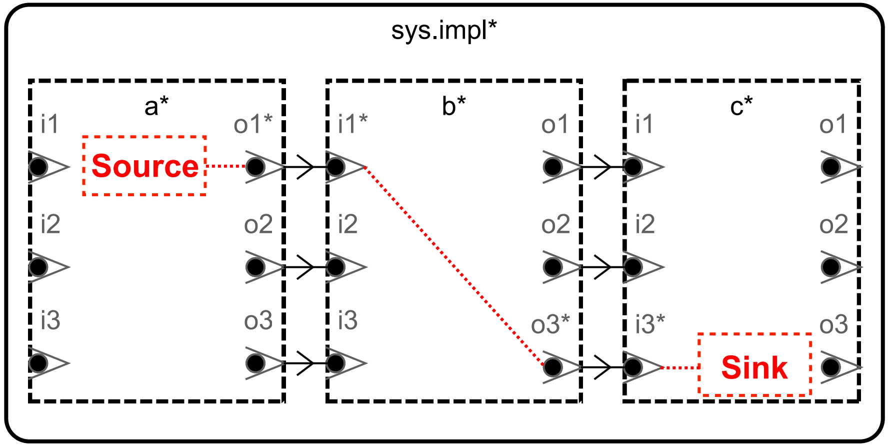
Context (2): AADL Error Modeling Annex

EMV2

Extension of core AADL for modeling off-nominal behavior

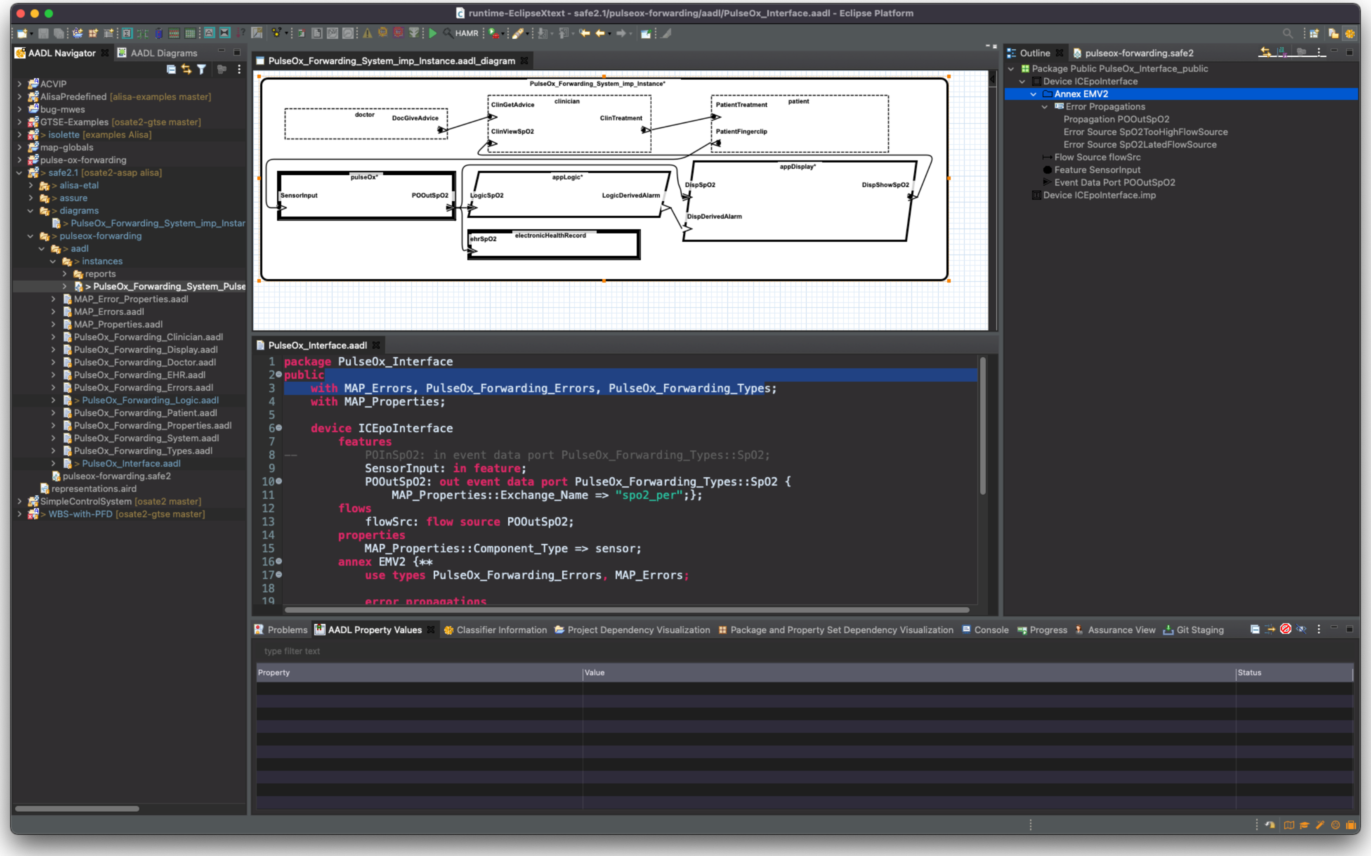
We rely heavily on:

- Error Types
- Propagation Paths
- Error Flows
  - Creation
  - Propagation
  - Transformation
  - Consumption



Context (3): Open Source AADL Tool Environment

OSATE


runtime-EclipseXtext - safe2.1/pulseox-forwarding/aadl/PulseOx_Interface.aadl - Eclipse Platform
HAMR
AADL Navigator
AADL Diagrams
ACVIP
AlisaPredefined [alisa-examples master]
bug-mwes
GTSE-Examples [osate2-gtse master]
> isolette [examples Alisa]
map-globals
pulse-ox-forwarding
> safe2.1 [osate2-asap alisa]
> alisa-etal
> assure
> diagrams
> PulseOx_Forwarding_System_imp_Instar
> pulseox-forwarding
> aadl
> instances
reports
> PulseOx_Forwarding_System_Pulse
MAP_Error_Properties.aadl
MAP_Errors.aadl
MAP_Properties.aadl
PulseOx_Forwarding_Clinician.aadl
PulseOx_Forwarding_Display.aadl
PulseOx_Forwarding_Doctor.aadl
PulseOx_Forwarding_EHR.aadl
PulseOx_Forwarding_Errors.aadl
> PulseOx_Forwarding_Logic.aadl
PulseOx_Forwarding_Patient.aadl
PulseOx_Forwarding_Properties.aadl
PulseOx_Forwarding_System.aadl
PulseOx_Forwarding_Types.aadl
> PulseOx_Interface.aadl
pulseox-forwarding.safe2
representations.aird
SimpleControlSystem [osate2 master]
> WBS-with-PFD [osate2-gtse master]
PulseOx_Forwarding_System_imp_Instance.aadl_diagram
PulseOx_Forwarding_System_imp_Instance*
doctor
DocGiveAdvice
clinician
ClinGetAdvice
ClinViewSpO2
ClinTreatment
patient
PatientTreatment
PatientFingerclip
pulseOx*
SensorInput
POOutSpO2
appLogic*
LogicSpO2
LogicDerivedAlarm
appDisplay*
DispSpO2
DispDerivedAlarm
DispShowSpO2
electronicHealthRecord
ehrSpO2
PulseOx_Interface.aadl
Outline
pulseox-forwarding.safe2
Package Public PulseOx_Interface_public
Device ICEpoInterface
Annex EMV2
Error Propagations
Propagation POOutSpO2
Error Source SpO2TooHighFlowSource
Error Source SpO2LatedFlowSource
Flow Source flowSrc
Feature SensorInput
Event Data Port POOutSpO2
Device ICEpoInterface.imp
Problems
AADL Property Values
Classifier Information
Project Dependency Visualization
Package and Property Set Dependency Visualization
Console
Progress
Assurance View
Git Staging
type filter text
Property
Value
Status

```
package PulseOx_Interface
public
    with MAP_Errors, PulseOx_Forwarding_Errors, PulseOx_Forwarding_Types;
    with MAP_Properties;

    device ICEpoInterface
        features
--          POInSpO2: in event data port PulseOx_Forwarding_Types::SpO2;
            SensorInput: in feature;
            POOutSpO2: out event data port PulseOx_Forwarding_Types::SpO2 {
                MAP_Properties::Exchange_Name => "spo2_per";};
        flows
            flowSrc: flow source POOutSpO2;
        properties
            MAP_Properties::Component_Type => sensor;
        annex EMV2 {**
            use types PulseOx_Forwarding_Errors, MAP_Errors;

            error propagations
```

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Context (4): Program and Model Slicing

1. Weiser's Original Concept: Reduced, but still executable, version of program [1]
2. Ahmadi's Slicer: Reduced model (UML-RT) preserving structural and behavioral aspects [2]
3. SafeSlice: Requirement traceability across multiple levels of safety-critical system models (SysML) [3]
4. Kompren: Combines metamodel and conformant model to generate a slicer [4]
5. Awas: OSATE plugin for calculating reachability queries across AADL models [5]

[1] "Program Slicing." Mark Weiser. IEEE Transactions on Software Engineering, 1984.

[2] "Slicing UML-based Models of Real-time Embedded Systems." Reza Ahmadi, Ernesto Posse, Juergen Dingel. MODELS, 2018.

[3] "Traceability and SysML Design Slices to Support Safety Inspections: A Controlled Experiment." Lionel Briand, Davide Falessi, Shiva Nejati, Mehrdad Sabetzadeh, Tao Yue. ACM Trans. on SW Eng and Methodology, 2014.

[4] "Kompren: Modeling and Generating Model Slicers." Arnaud Blouin, Benoît Combemale, Benoit Baudry, Olivier Beaudoux. Software & Systems Modeling, 2012.

[5] "Awas: AADL Information Flow and Error Propagation Analysis Framework." Hariharan Thiagarajan, John Hatcliff, Robby. Innovations in Systems and Software Engineering, 2022.

Solution: The *OSATE Slicer*

Goal: Reachability calculations built into OSATE that are…

- Usable
  - Data and control flow are used in a number of analyses
  - … so the Slicer should be too
- Maintainable
  - Align with existing OSATE design principles, tooling, and infrastructure
- Performant
  - Reduced installation complexity and execution time

Agenda

- Introduction & Background
- **The OSATE Slicer**
  - **Representation**
  - **Generation**
  - **Queries**
- Evaluation

Graph Representation

Slicer generates and queries two graphs

- Nominal (Core AADL)
- Off-Nominal (Core AADL + EMV2)

Nominal:

1. $\mathcal{G}_N = (V_N, \rightarrow_{e_N})$
   1. $V_N = F \cup A_{used}$
   2. $\rightarrow_{e_N} \subseteq V_N \times V_N$

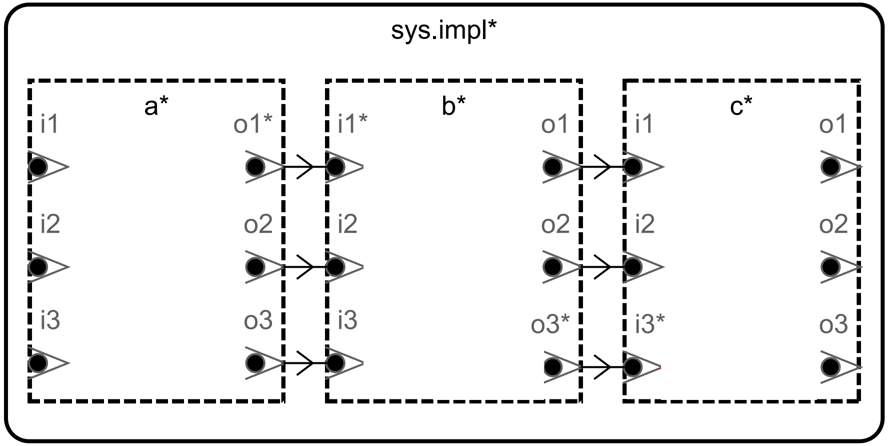
Off-Nominal:

2. $\mathcal{G}_O = \left(V_O, \rightarrow_{e_O}\right)$
   1. $V_O = L \times T$
   2. $L = P_F \times P_B \times P_A \times P_P \times R_{Src} \times R_{Snk}$
   3. $\rightarrow_{e_O} \subseteq V_O \times V_O$

Graph Generation

Nominal

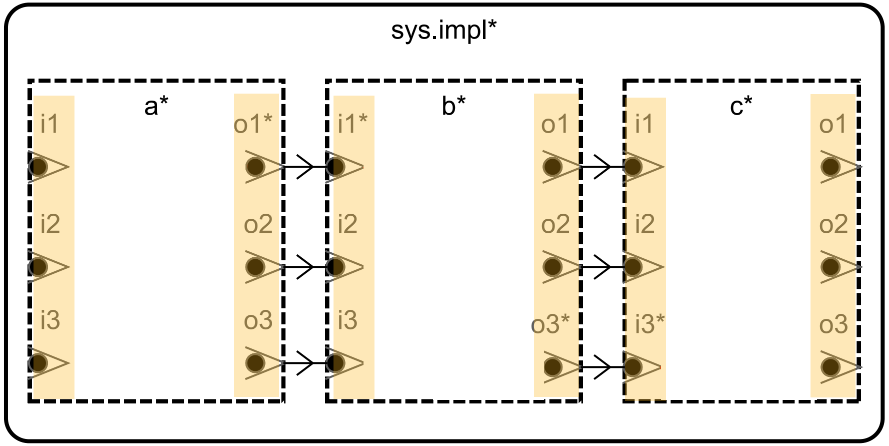
(a) Nominal Reachability

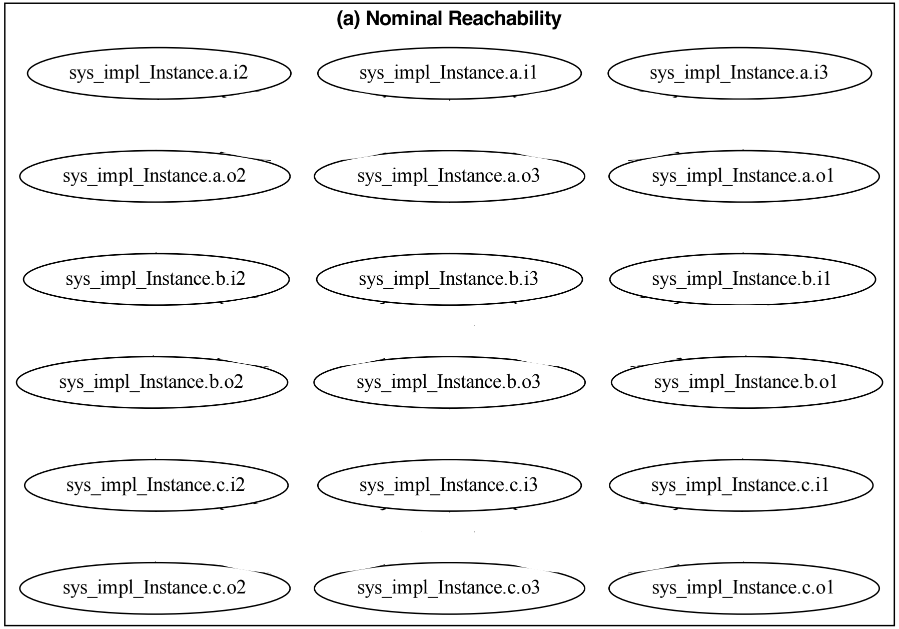


```
function ConstructNominal($m$)
    $ED := \emptyset$ ▷ Components with explicit decompositions
    $\mathcal{G}_N = (V_N, \rightarrow_{e_N}) := (\emptyset, \emptyset)$ ▷ Initialize $\mathcal{G}_N$
    for $feat \in m$ do ▷ $feat$ is a feature
        $V_N := V_N \cup \textsc{Vert}(feat)$
    for $conn \in m$ do ▷ $conn$ is a connection
        $V_N := V_N \cup \textsc{Vert}(conn_{Src}) \cup \textsc{Vert}(conn_{Dst})$
        $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(conn_{Src}, conn_{Dst})$
        $ED := ED \cup \textsc{Container}(conn)$
    for $flow \in m$ do ▷ $flow$ is a end to end flow
        $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(flow_{Src}, flow_{Dst})$
        $ED := ED \cup \textsc{Container}(flow)$
    for $comp \in (m \setminus ED)$ do ▷ $comp$ is a component
        for $feat_{in} \in comp$ do
            for $feat_{out} \in comp$ do
                $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(feat_{in}, feat_{out})$
```

Graph Generation

Nominal



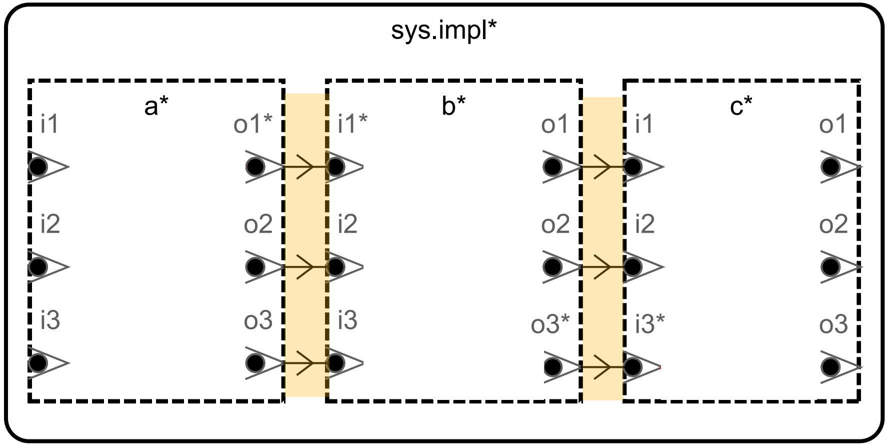


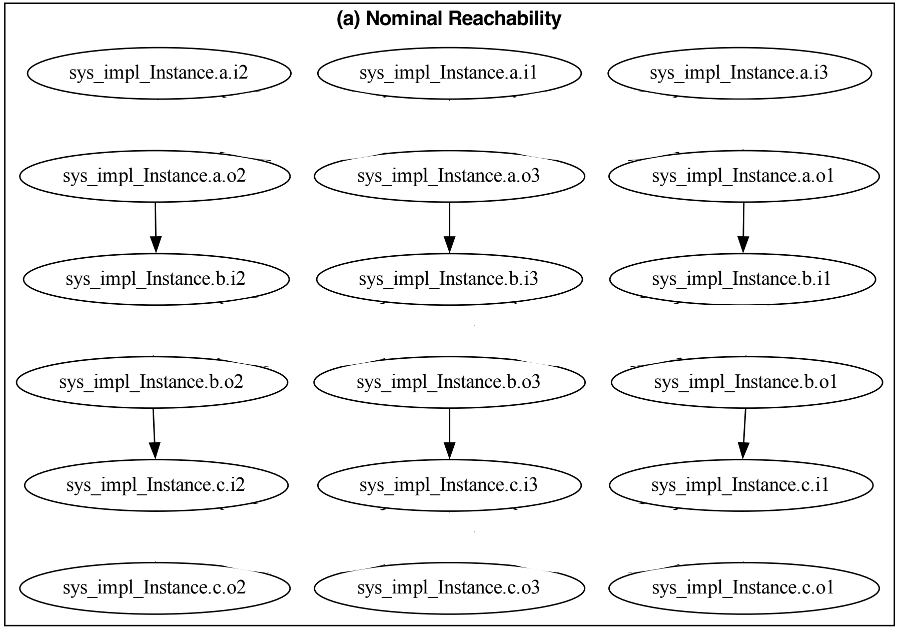
(a) Nominal Reachability

```
function ConstructNominal($m$)
    $ED := \emptyset$ ▷ Components with explicit decompositions
    $\mathcal{G}_N = (V_N, \rightarrow_{e_N}) := (\emptyset, \emptyset)$ ▷ Initialize $\mathcal{G}_N$
    for $feat \in m$ do ▷ $feat$ is a feature
        $V_N := V_N \cup \text{Vert}(feat)$
    for $conn \in m$ do ▷ $conn$ is a connection
        $V_N := V_N \cup \text{Vert}(conn_{Src}) \cup \text{Vert}(conn_{Dst})$
        $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \text{Edge}(conn_{Src}, conn_{Dst})$
        $ED := ED \cup \text{Container}(conn)$
    for $flow \in m$ do ▷ $flow$ is a end to end flow
        $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \text{Edge}(flow_{Src}, flow_{Dst})$
        $ED := ED \cup \text{Container}(flow)$
    for $comp \in (m \setminus ED)$ do ▷ $comp$ is a component
        for $feat_{in} \in comp$ do
            for $feat_{out} \in comp$ do
                $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \text{Edge}(feat_{in}, feat_{out})$
```

Graph Generation

Nominal

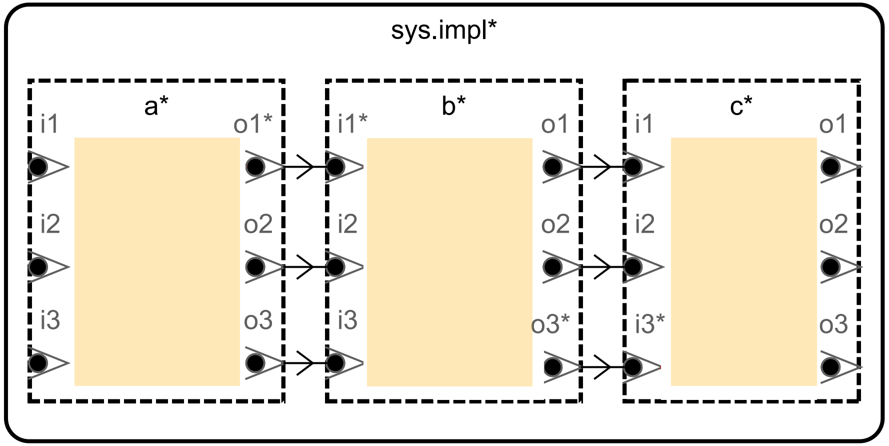


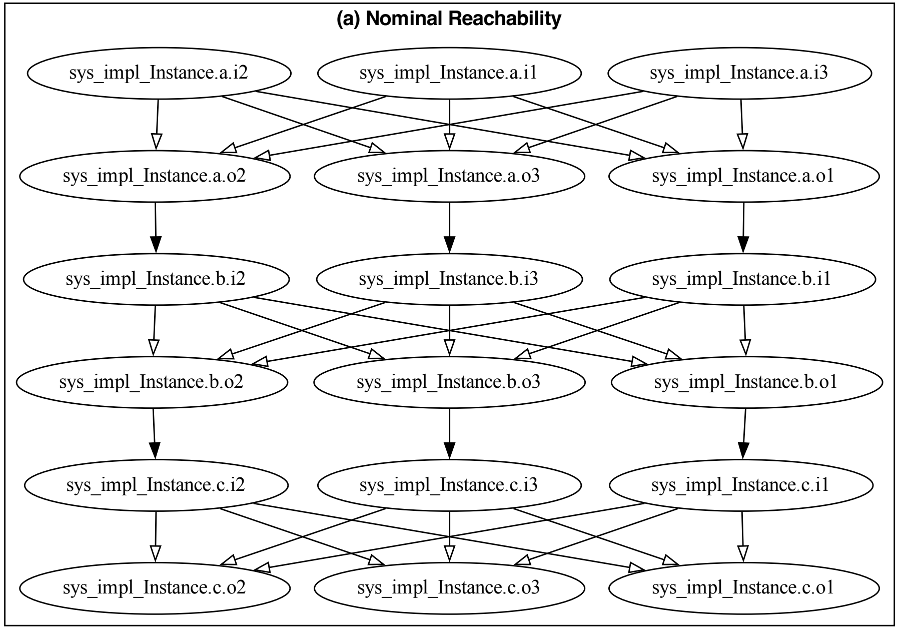

(a) Nominal Reachability

```
function $\textsc{ConstructNominal}(m)$
    $ED := \emptyset$ ▷ Components with explicit decompositions
    $\mathcal{G}_N = (V_N, \rightarrow_{e_N}) := (\emptyset, \emptyset)$ ▷ Initialize $\mathcal{G}_N$
    for $feat \in m$ do ▷ $feat$ is a feature
        $V_N := V_N \cup \textsc{Vert}(feat)$
    for $conn \in m$ do ▷ $conn$ is a connection
        $V_N := V_N \cup \textsc{Vert}(conn_{Src}) \cup \textsc{Vert}(conn_{Dst})$
        $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(conn_{Src}, conn_{Dst})$
        $ED := ED \cup \textsc{Container}(conn)$
    for $flow \in m$ do ▷ $flow$ is a end to end flow
        $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(flow_{Src}, flow_{Dst})$
        $ED := ED \cup \textsc{Container}(flow)$
    for $comp \in (m \setminus ED)$ do ▷ $comp$ is a component
        for $feat_{in} \in comp$ do
            for $feat_{out} \in comp$ do
                $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(feat_{in}, feat_{out})$
```

Graph Generation

Nominal



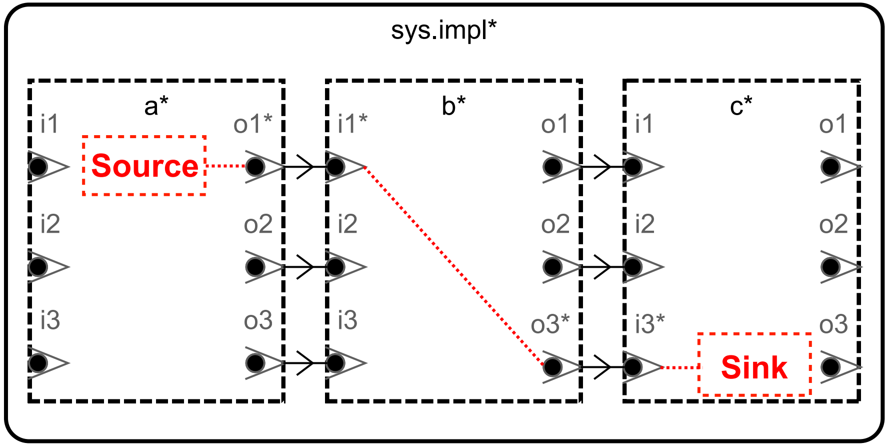


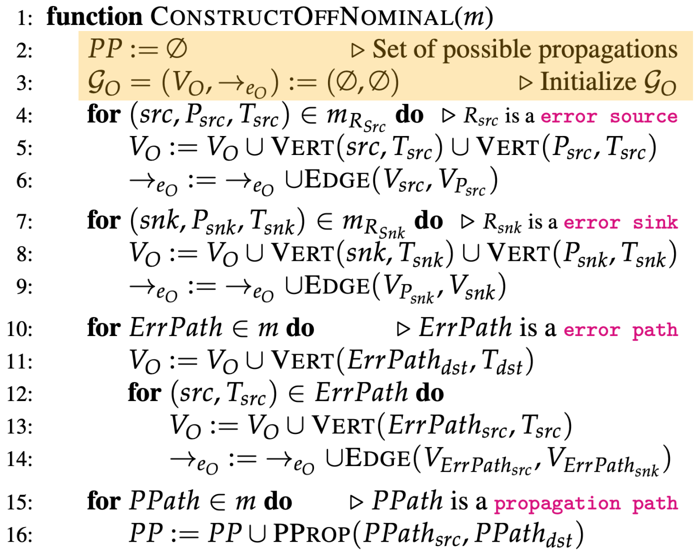
```
function CONSTRUCTNOMINAL($m$)
    $ED := \varnothing$ ▷ Components with explicit decompositions
    $\mathcal{G}_N = (V_N, \rightarrow_{e_N}) := (\varnothing, \varnothing)$ ▷ Initialize $\mathcal{G}_N$
    for $feat \in m$ do ▷ $feat$ is a feature
        $V_N := V_N \cup \textsc{Vert}(feat)$
    for $conn \in m$ do ▷ $conn$ is a connection
        $V_N := V_N \cup \textsc{Vert}(conn_{Src}) \cup \textsc{Vert}(conn_{Dst})$
        $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(conn_{Src}, conn_{Dst})$
        $ED := ED \cup \textsc{Container}(conn)$
    for $flow \in m$ do ▷ $flow$ is a end to end flow
        $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(flow_{Src}, flow_{Dst})$
        $ED := ED \cup \textsc{Container}(flow)$
    for $comp \in (m \setminus ED)$ do ▷ $comp$ is a component
        for $feat_{in} \in comp$ do
            for $feat_{out} \in comp$ do
                $\rightarrow_{e_N} := \rightarrow_{e_N} \cup \textsc{Edge}(feat_{in}, feat_{out})$
```

Graph Generation

Off-Nominal

(b) Off-Nominal Reachability

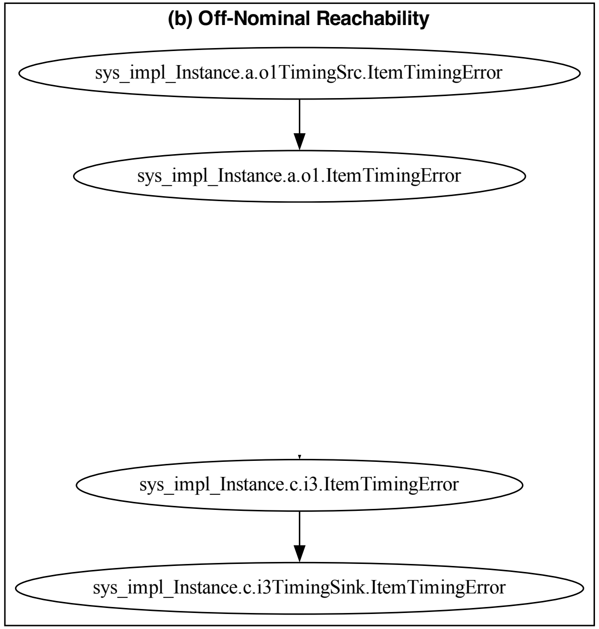




```
function CONSTRUCTOFFNOMINAL($m$)
    $PP := \emptyset$                                   ▷ Set of possible propagations
    $\mathcal{G}_O = (V_O, \rightarrow_{e_O}) := (\emptyset, \emptyset)$                ▷ Initialize $\mathcal{G}_O$
    for $(src, P_{src}, T_{src}) \in m_{R_{Src}}$ do   ▷ $R_{src}$ is a error source
        $V_O := V_O \cup \text{VERT}(src, T_{src}) \cup \text{VERT}(P_{src}, T_{src})$
        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \text{EDGE}(V_{src}, V_{P_{src}})$
    for $(snk, P_{snk}, T_{snk}) \in m_{R_{Snk}}$ do   ▷ $R_{snk}$ is a error sink
        $V_O := V_O \cup \text{VERT}(snk, T_{snk}) \cup \text{VERT}(P_{snk}, T_{snk})$
        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \text{EDGE}(V_{P_{snk}}, V_{snk})$
    for $ErrPath \in m$ do            ▷ $ErrPath$ is a error path
        $V_O := V_O \cup \text{VERT}(ErrPath_{dst}, T_{dst})$
        for $(src, T_{src}) \in ErrPath$ do
            $V_O := V_O \cup \text{VERT}(ErrPath_{src}, T_{src})$
            $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \text{EDGE}(V_{ErrPath_{src}}, V_{ErrPath_{snk}})$
    for $PPath \in m$ do          ▷ $PPath$ is a propagation path
        $PP := PP \cup \text{PPROP}(PPath_{src}, PPath_{dst})$
```

Graph Generation

Off-Nominal



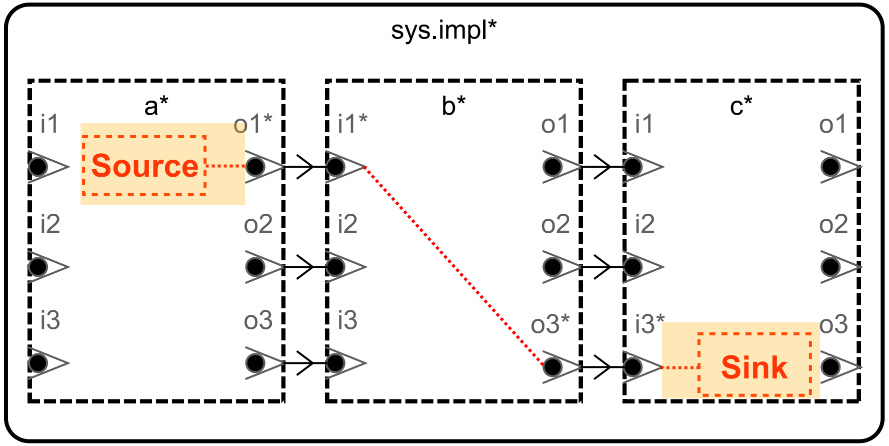
(b) Off-Nominal Reachability

sys_impl_Instance.a.o1TimingSrc.ItemTimingError

sys_impl_Instance.a.o1.ItemTimingError

sys_impl_Instance.c.i3.ItemTimingError

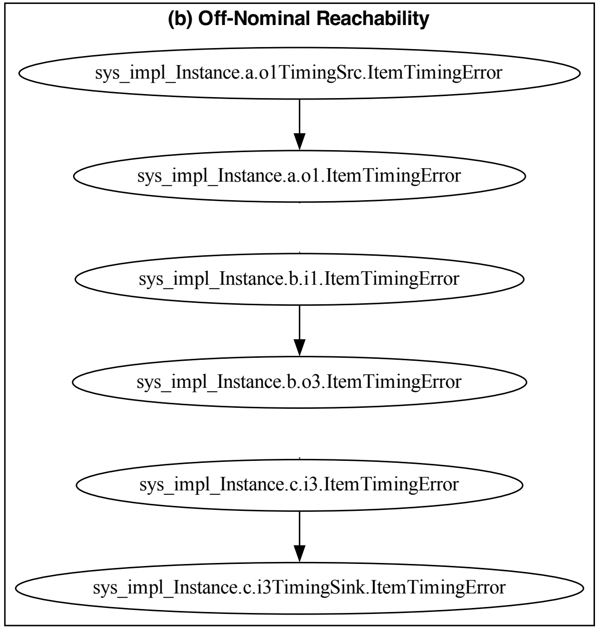
sys_impl_Instance.c.i3TimingSink.ItemTimingError



```
function ConstructOffNominal($m$)
    $PP := \emptyset$    ▷ Set of possible propagations
    $\mathcal{G}_O = (V_O, \rightarrow_{e_O}) := (\emptyset, \emptyset)$    ▷ Initialize $\mathcal{G}_O$
    for $(src, P_{src}, T_{src}) \in m_{R_{Src}}$ do    ▷ $R_{src}$ is a error source
        $V_O := V_O \cup \textsc{Vert}(src, T_{src}) \cup \textsc{Vert}(P_{src}, T_{src})$
        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \textsc{Edge}(V_{src}, V_{P_{src}})$
    for $(snk, P_{snk}, T_{snk}) \in m_{R_{Snk}}$ do    ▷ $R_{snk}$ is a error sink
        $V_O := V_O \cup \textsc{Vert}(snk, T_{snk}) \cup \textsc{Vert}(P_{snk}, T_{snk})$
        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \textsc{Edge}(V_{P_{snk}}, V_{snk})$
    for $ErrPath \in m$ do    ▷ $ErrPath$ is a error path
        $V_O := V_O \cup \textsc{Vert}(ErrPath_{dst}, T_{dst})$
        for $(src, T_{src}) \in ErrPath$ do
            $V_O := V_O \cup \textsc{Vert}(ErrPath_{src}, T_{src})$
            $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \textsc{Edge}(V_{ErrPath_{src}}, V_{ErrPath_{snk}})$
    for $PPath \in m$ do    ▷ $PPath$ is a propagation path
        $PP := PP \cup \textsc{PProp}(PPath_{src}, PPath_{dst})$
```

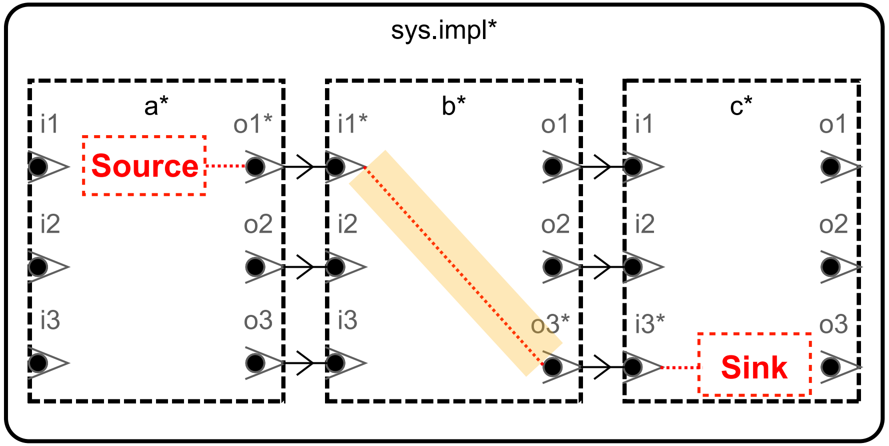
Graph Generation

Off-Nominal



(b) Off-Nominal Reachability

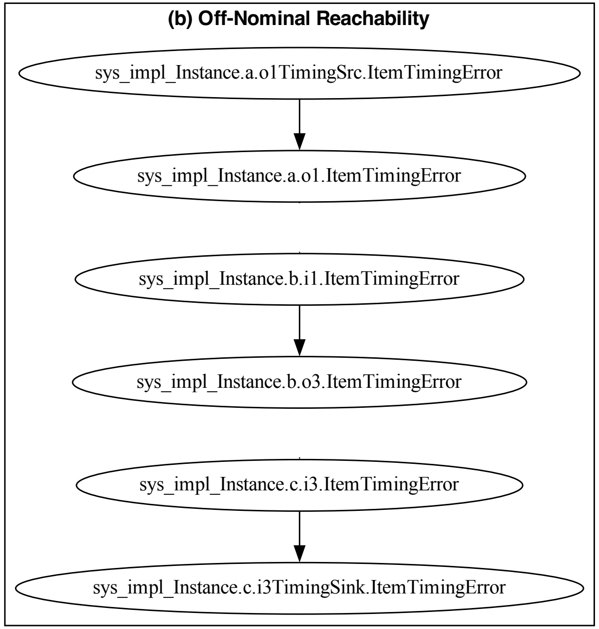
sys_impl_Instance.a.o1TimingSrc.ItemTimingError
sys_impl_Instance.a.o1.ItemTimingError
sys_impl_Instance.b.i1.ItemTimingError
sys_impl_Instance.b.o3.ItemTimingError
sys_impl_Instance.c.i3.ItemTimingError
sys_impl_Instance.c.i3TimingSink.ItemTimingError

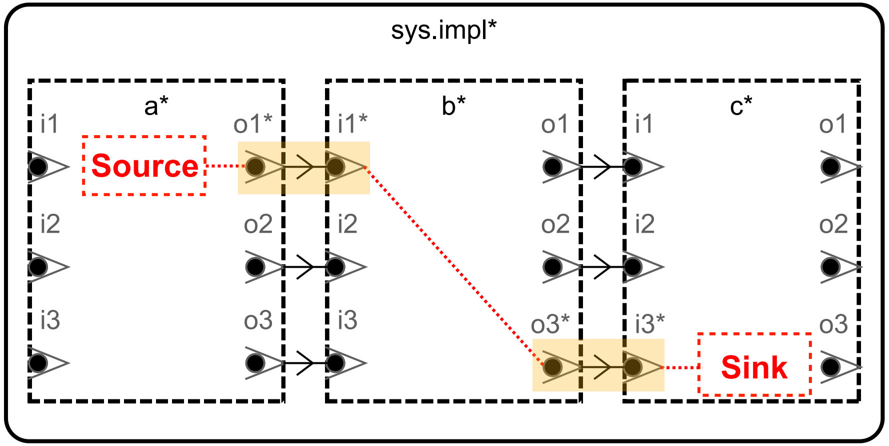


```
function ConstructOffNominal($m$)
    $PP := \emptyset$    ▷ Set of possible propagations
    $\mathcal{G}_O = (V_O, \rightarrow_{e_O}) := (\emptyset, \emptyset)$    ▷ Initialize $\mathcal{G}_O$
    for $(src, P_{src}, T_{src}) \in m_{R_{Src}}$ do    ▷ $R_{src}$ is a error source
        $V_O := V_O \cup \textsc{Vert}(src, T_{src}) \cup \textsc{Vert}(P_{src}, T_{src})$
        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \textsc{Edge}(V_{src}, V_{P_{src}})$
    for $(snk, P_{snk}, T_{snk}) \in m_{R_{Snk}}$ do    ▷ $R_{snk}$ is a error sink
        $V_O := V_O \cup \textsc{Vert}(snk, T_{snk}) \cup \textsc{Vert}(P_{snk}, T_{snk})$
        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \textsc{Edge}(V_{P_{snk}}, V_{snk})$
    for $ErrPath \in m$ do    ▷ $ErrPath$ is a error path
        $V_O := V_O \cup \textsc{Vert}(ErrPath_{dst}, T_{dst})$
        for $(src, T_{src}) \in ErrPath$ do
            $V_O := V_O \cup \textsc{Vert}(ErrPath_{src}, T_{src})$
            $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \textsc{Edge}(V_{ErrPath_{src}}, V_{ErrPath_{snk}})$
    for $PPath \in m$ do    ▷ $PPath$ is a propagation path
        $PP := PP \cup \textsc{PProp}(PPath_{src}, PPath_{dst})$
```

Graph Generation

Off-Nominal

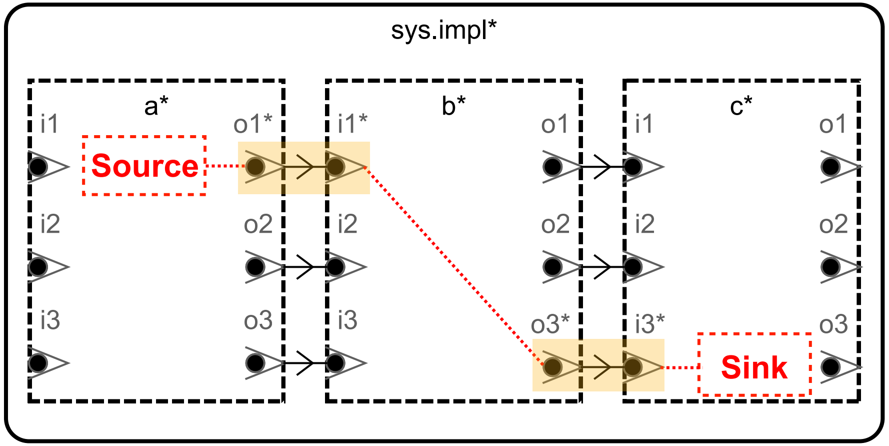

(b) Off-Nominal Reachability
sys_impl_Instance.a.o1TimingSrc.ItemTimingError
sys_impl_Instance.a.o1.ItemTimingError
sys_impl_Instance.b.i1.ItemTimingError
sys_impl_Instance.b.o3.ItemTimingError
sys_impl_Instance.c.i3.ItemTimingError
sys_impl_Instance.c.i3TimingSink.ItemTimingError

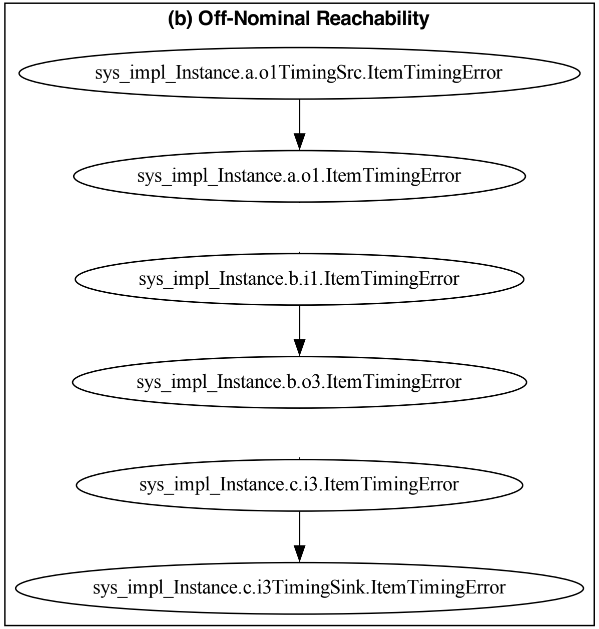


```
function CONSTRUCTOFFNOMINAL($m$)
    $PP := \emptyset$    ▷ Set of possible propagations
    $\mathcal{G}_O = (V_O, \rightarrow_{e_O}) := (\emptyset, \emptyset)$    ▷ Initialize $\mathcal{G}_O$
    for $(src, P_{src}, T_{src}) \in m_{R_{Src}}$ do    ▷ $R_{src}$ is a error source
        $V_O := V_O \cup \text{VERT}(src, T_{src}) \cup \text{VERT}(P_{src}, T_{src})$
        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \text{EDGE}(V_{src}, V_{P_{src}})$
    for $(snk, P_{snk}, T_{snk}) \in m_{R_{Snk}}$ do    ▷ $R_{snk}$ is a error sink
        $V_O := V_O \cup \text{VERT}(snk, T_{snk}) \cup \text{VERT}(P_{snk}, T_{snk})$
        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \text{EDGE}(V_{P_{snk}}, V_{snk})$
    for $ErrPath \in m$ do    ▷ $ErrPath$ is a error path
        $V_O := V_O \cup \text{VERT}(ErrPath_{dst}, T_{dst})$
        for $(src, T_{src}) \in ErrPath$ do
            $V_O := V_O \cup \text{VERT}(ErrPath_{src}, T_{src})$
            $\rightarrow_{e_O} := \rightarrow_{e_O} \cup \text{EDGE}(V_{ErrPath_{src}}, V_{ErrPath_{snk}})$
    for $PPath \in m$ do    ▷ $PPath$ is a propagation path
        $PP := PP \cup \text{PPROP}(PPath_{src}, PPath_{dst})$
```

Graph Generation

Off-Nominal – Fixpoint Calculation





(b) Off-Nominal Reachability

sys_impl_Instance.a.o1TimingSrc.ItemTimingError

sys_impl_Instance.a.o1.ItemTimingError

sys_impl_Instance.b.i1.ItemTimingError

sys_impl_Instance.b.o3.ItemTimingError

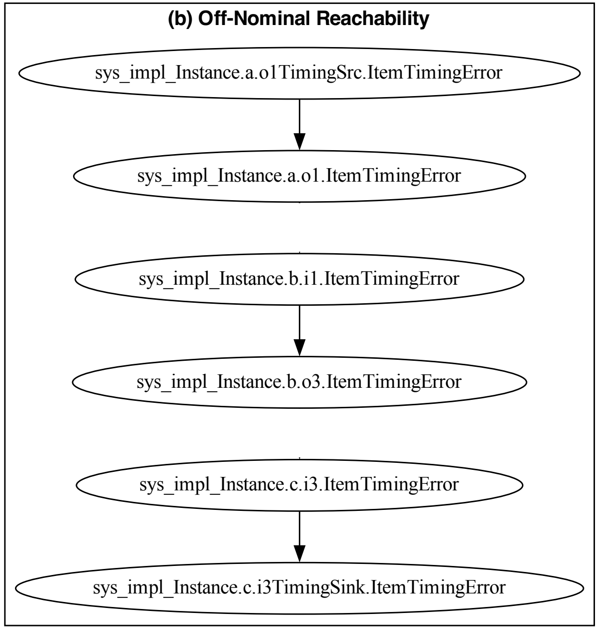
sys_impl_Instance.c.i3.ItemTimingError

sys_impl_Instance.c.i3TimingSink.ItemTimingError

```
function CALCULATEFIXPOINT($R_{Src}, V_O, \rightarrow_{e_O}, PP$)
    repeat
        $\rightarrow'_{e_O} := \rightarrow_{e_O}$
        for $src \in R_{Src}$ do                    ▷ $src$ is a error source
            $edges := V_{src_{Out}}$               ▷ Outgoing edges of $V_{src}$
            while $|edges| > 0$ do
                $CurrEdge :=$ POP$(edges)$
                $src := V_{CurrEdge_{Dst}}$        ▷ $CurrEdge$'s dest.
                for $OutEdge \in src_{Out}$ do
                    $edges := edges \cup OutEdge$
                for $Prop \in \{PP|Src = src\}$ do
                    $tgt := Prop_{Dst}$
                    $NewEdge :=$ EDGE$(src, tgt)$
                    if $NewEdge \notin \rightarrow_{e_O}$ then
                        $edges := edges \cup NewEdge$
                        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup NewEdge$
    until $\rightarrow'_{e_O} = \rightarrow_{e_O}$   ▷ Halt when edge set is unmodified
```

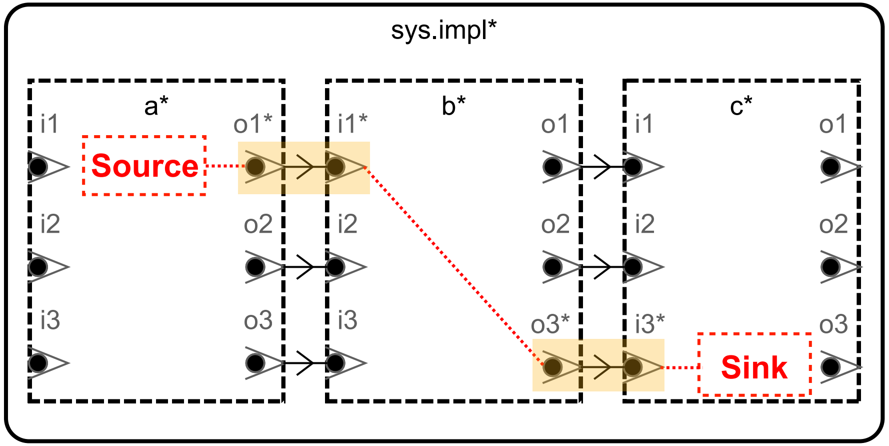
Graph Generation

Off-Nominal – Fixpoint Calculation



(b) Off-Nominal Reachability

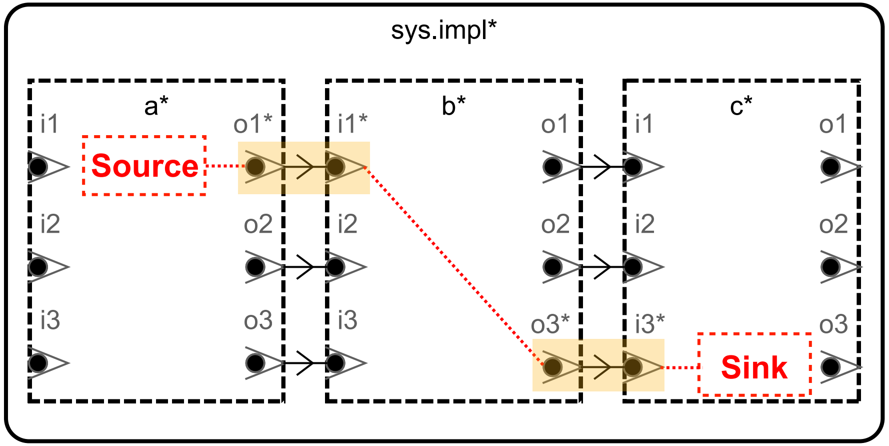
sys_impl_Instance.a.o1TimingSrc.ItemTimingError
sys_impl_Instance.a.o1.ItemTimingError
sys_impl_Instance.b.i1.ItemTimingError
sys_impl_Instance.b.o3.ItemTimingError
sys_impl_Instance.c.i3.ItemTimingError
sys_impl_Instance.c.i3TimingSink.ItemTimingError

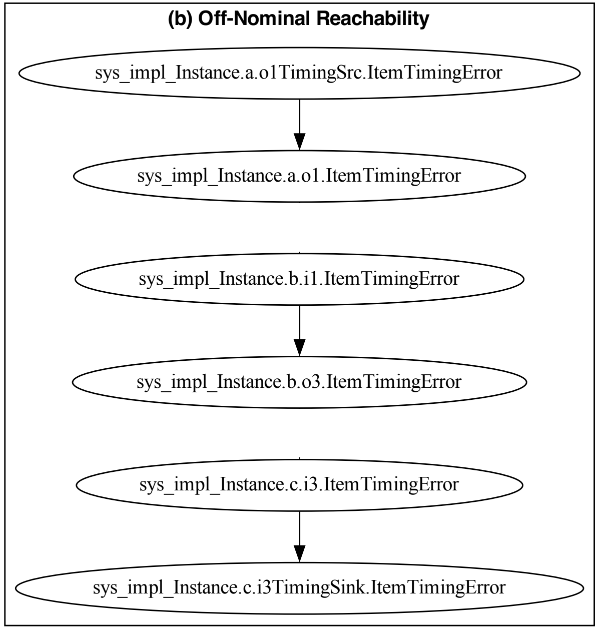


```
function CalculateFixpoint($R_{Src}, V_O, \rightarrow_{e_O}, PP$)
    repeat
        $\rightarrow'_{e_O} := \rightarrow_{e_O}$
        for $src \in R_{Src}$ do                          ▷ $src$ is a error source
            $edges := V_{src_{Out}}$                      ▷ Outgoing edges of $V_{src}$
            while $|edges| > 0$ do
                $CurrEdge := \text{Pop}(edges)$
                $src := V_{CurrEdge_{Dst}}$               ▷ $CurrEdge$'s dest.
                for $OutEdge \in src_{Out}$ do
                    $edges := edges \cup OutEdge$
                for $Prop \in \{PP|Src = src\}$ do
                    $tgt := Prop_{Dst}$
                    $NewEdge := \text{Edge}(src, tgt)$
                    if $NewEdge \notin \rightarrow_{e_O}$ then
                        $edges := edges \cup NewEdge$
                        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup NewEdge$
    until $\rightarrow'_{e_O} = \rightarrow_{e_O}$    ▷ Halt when edge set is unmodified
```

Graph Generation

Off-Nominal – Fixpoint Calculation



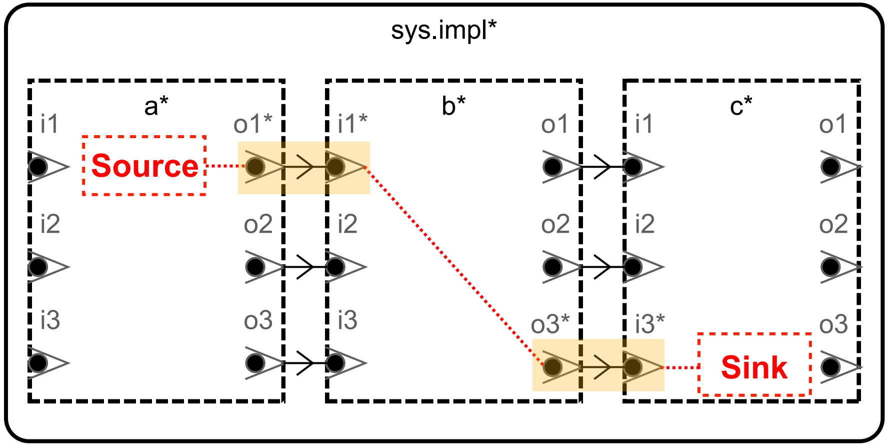

sys_impl_Instance.a.o1TimingSrc.ItemTimingError
sys_impl_Instance.a.o1.ItemTimingError
sys_impl_Instance.b.i1.ItemTimingError
sys_impl_Instance.b.o3.ItemTimingError
sys_impl_Instance.c.i3.ItemTimingError
sys_impl_Instance.c.i3TimingSink.ItemTimingError

(b) Off-Nominal Reachability

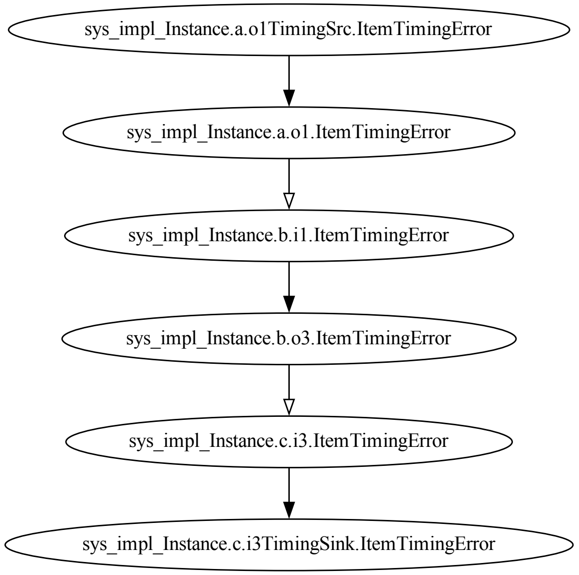
```
function CalculateFixpoint($R_{Src}, V_O, \rightarrow_{e_O}, PP$)
    repeat
        $\rightarrow'_{e_O} := \rightarrow_{e_O}$
        for $src \in R_{Src}$ do                    ▷ $src$ is a error source
            $edges := V_{src_{Out}}$                 ▷ Outgoing edges of $V_{src}$
            while $|edges| > 0$ do
                $CurrEdge :=$ Pop$(edges)$
                $src := V_{CurrEdge_{Dst}}$          ▷ $CurrEdge$'s dest.
                for $OutEdge \in src_{Out}$ do
                    $edges := edges \cup OutEdge$
                for $Prop \in \{PP|Src = src\}$ do
                    $tgt := Prop_{Dst}$
                    $NewEdge :=$ Edge$(src, tgt)$
                    if $NewEdge \notin \rightarrow_{e_O}$ then
                        $edges := edges \cup NewEdge$
                        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup NewEdge$
    until $\rightarrow'_{e_O} = \rightarrow_{e_O}$    ▷ Halt when edge set is unmodified
```

Graph Generation

Off-Nominal – Fixpoint Calculation



(b) Off-Nominal Reachability

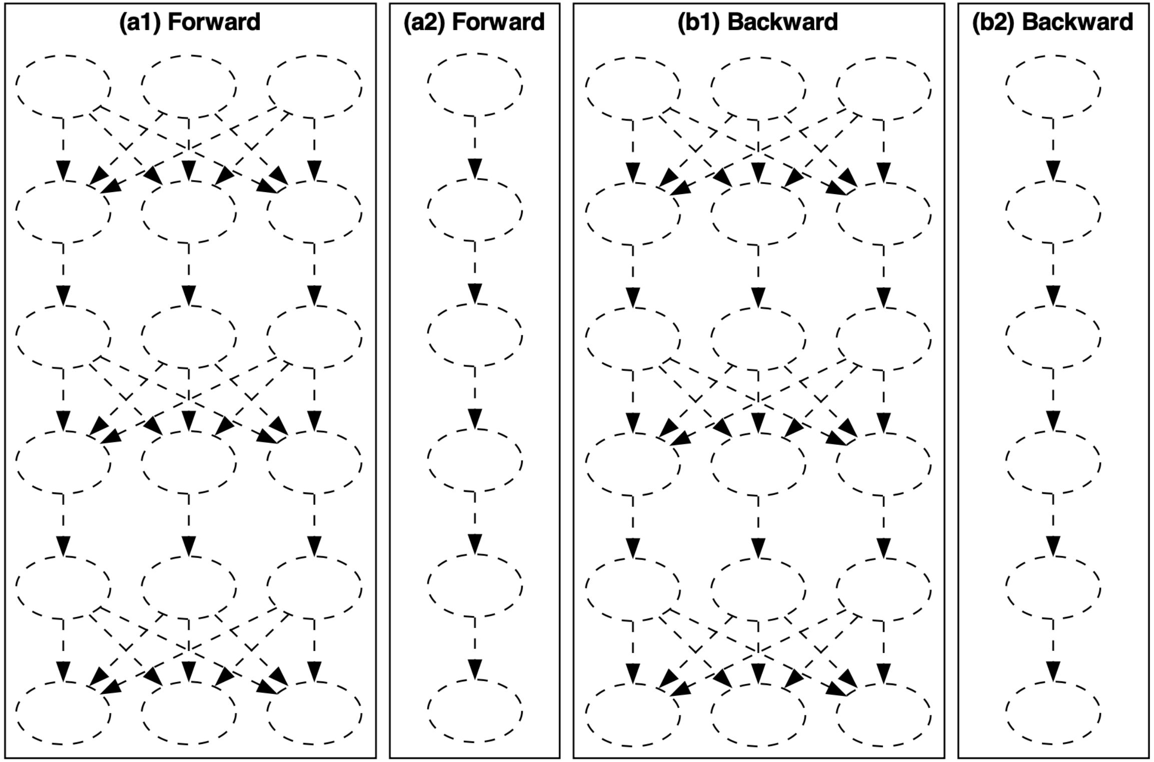


sys_impl_Instance.a.o1TimingSrc.ItemTimingError
sys_impl_Instance.a.o1.ItemTimingError
sys_impl_Instance.b.i1.ItemTimingError
sys_impl_Instance.b.o3.ItemTimingError
sys_impl_Instance.c.i3.ItemTimingError
sys_impl_Instance.c.i3TimingSink.ItemTimingError

```
function CalculateFixpoint($R_{Src}, V_O, \rightarrow_{e_O}, PP$)
    repeat
        $\rightarrow'_{e_O} := \rightarrow_{e_O}$
        for $src \in R_{Src}$ do                    ▷ $src$ is a error source
            $edges := V_{src_{Out}}$                ▷ Outgoing edges of $V_{src}$
            while $|edges| > 0$ do
                $CurrEdge :=$ Pop$(edges)$
                $src := V_{CurrEdge_{Dst}}$          ▷ $CurrEdge$'s dest.
                for $OutEdge \in src_{Out}$ do
                    $edges := edges \cup OutEdge$
                for $Prop \in \{PP|Src = src\}$ do
                    $tgt := Prop_{Dst}$
                    $NewEdge :=$ Edge$(src, tgt)$
                    if $NewEdge \notin \rightarrow_{e_O}$ then
                        $edges := edges \cup NewEdge$
                        $\rightarrow_{e_O} := \rightarrow_{e_O} \cup NewEdge$
    until $\rightarrow'_{e_O} = \rightarrow_{e_O}$      ▷ Halt when edge set is unmodified
```

Graph Queries

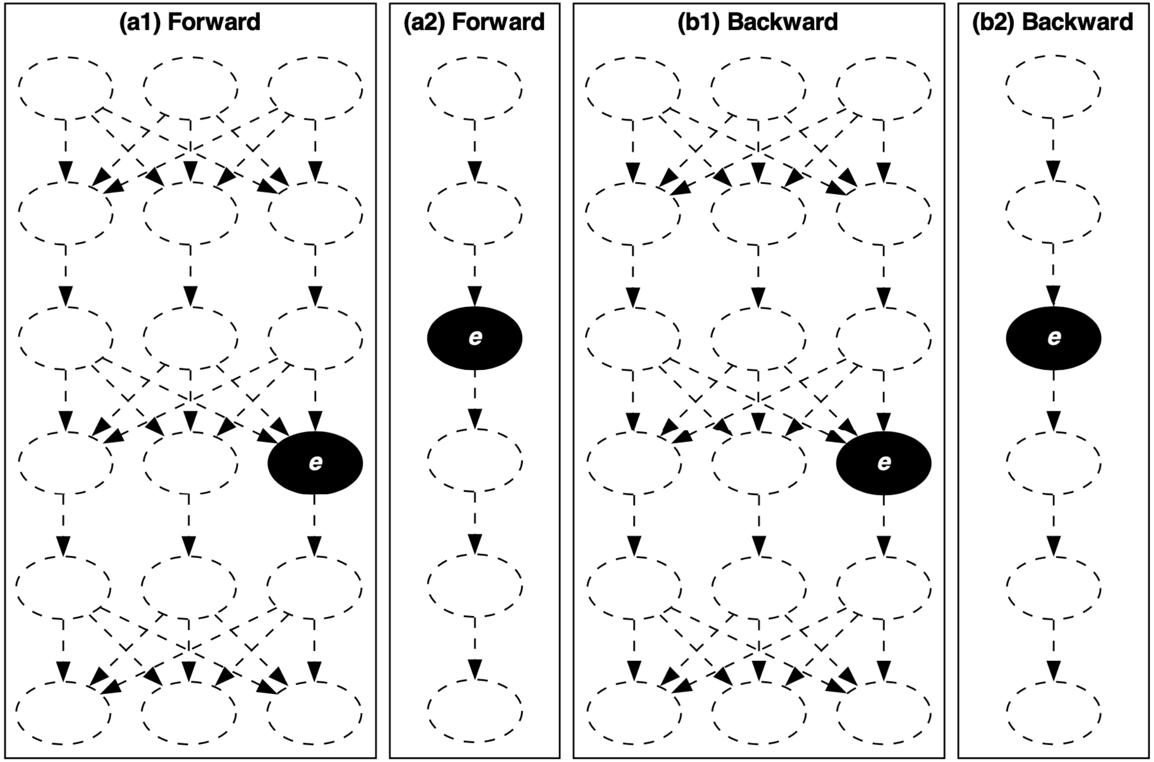
Forward & Backward Reach



```
function $\textsc{Reach}(\mathcal{G}, v_{origin})$
    $\mathcal{G}_{sub} = (V_{sub}, \rightarrow_{e_{sub}}) := (\emptyset, \emptyset) = \textsc{SubGraph}(\mathcal{G})$
    $V_{sub} := V_{sub} \cup v_{origin}$
    $v_{previous} := v_{origin}$
    for $v_{current} \in \textsc{BFIter}(\mathcal{G}, v_{origin})$ do
        $V_{sub} := V_{sub} \cup v_{current}$
        $\rightarrow_{e_{sub}} := \rightarrow_{e_{sub}} \cup \textsc{Edge}(v_{previous}, v_{current})$
        $v_{previous} := v_{current}$
```

Graph Queries

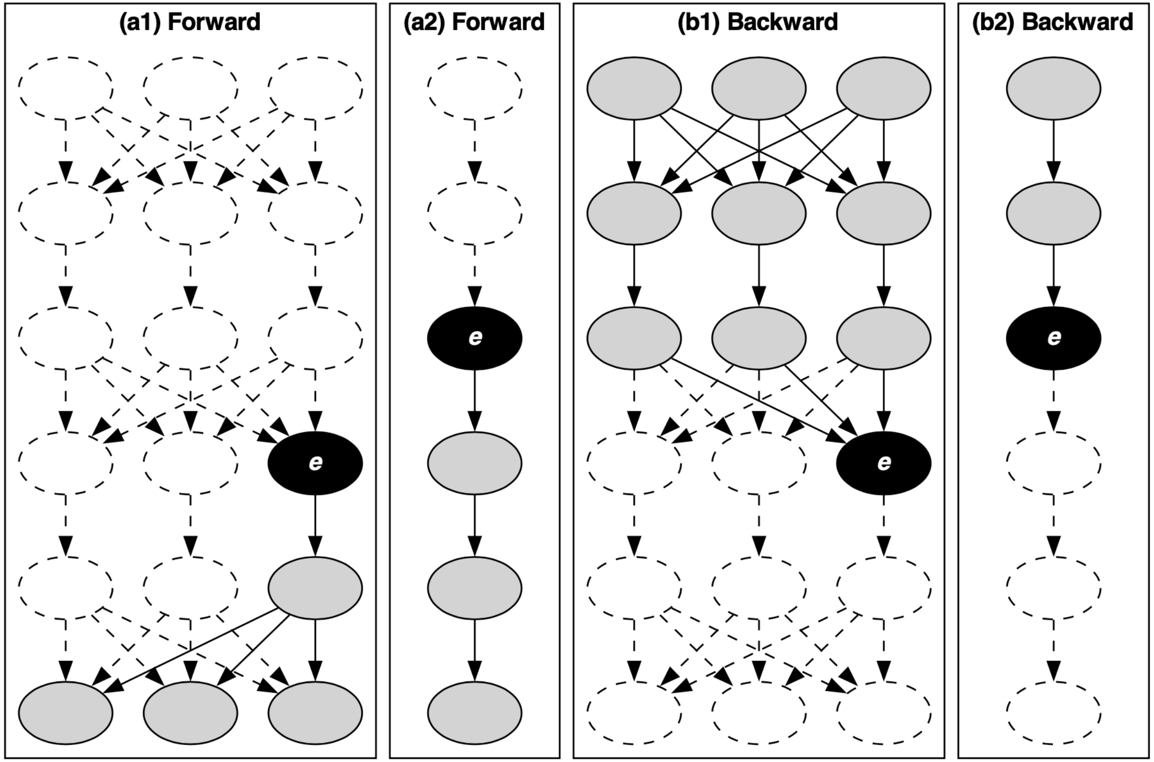
Forward & Backward Reach



```
function $\textsc{Reach}(\mathcal{G}, v_{origin})$
    $\mathcal{G}_{sub} = (V_{sub}, \rightarrow_{e_{sub}}) := (\emptyset, \emptyset) = \textsc{SubGraph}(\mathcal{G})$
    $V_{sub} := V_{sub} \cup v_{origin}$
    $v_{previous} := v_{origin}$
    for $v_{current} \in \textsc{BFIter}(\mathcal{G}, v_{origin})$ do
        $V_{sub} := V_{sub} \cup v_{current}$
        $\rightarrow_{e_{sub}} := \rightarrow_{e_{sub}} \cup \textsc{Edge}(v_{previous}, v_{current})$
        $v_{previous} := v_{current}$
```

Graph Queries

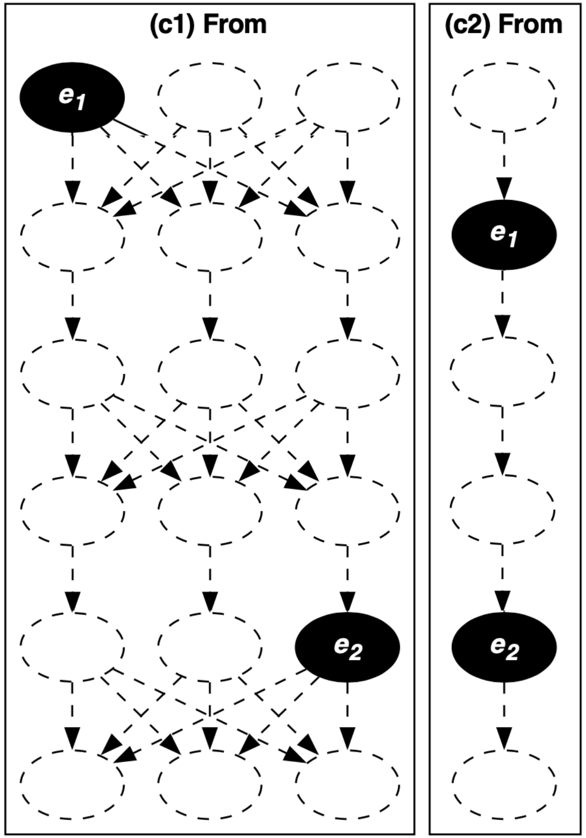
Forward & Backward Reach



```
function REACH($\mathcal{G}, v_{origin}$)
    $\mathcal{G}_{sub} = (V_{sub}, \rightarrow_{e_{sub}}) := (\emptyset, \emptyset) = \text{SUBGRAPH}(\mathcal{G})$
    $V_{sub} := V_{sub} \cup v_{origin}$
    $v_{previous} := v_{origin}$
    for $v_{current} \in \text{BFITER}(\mathcal{G}, v_{origin})$ do
        $V_{sub} := V_{sub} \cup v_{current}$
        $\rightarrow_{e_{sub}} := \rightarrow_{e_{sub}} \cup \text{EDGE}(v_{previous}, v_{current})$
        $v_{previous} := v_{current}$
```

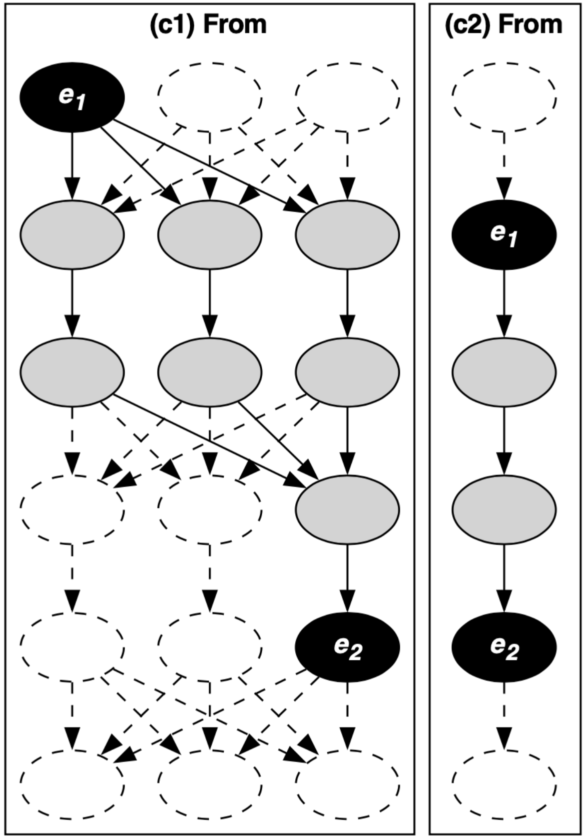
Graph Queries

Reach From



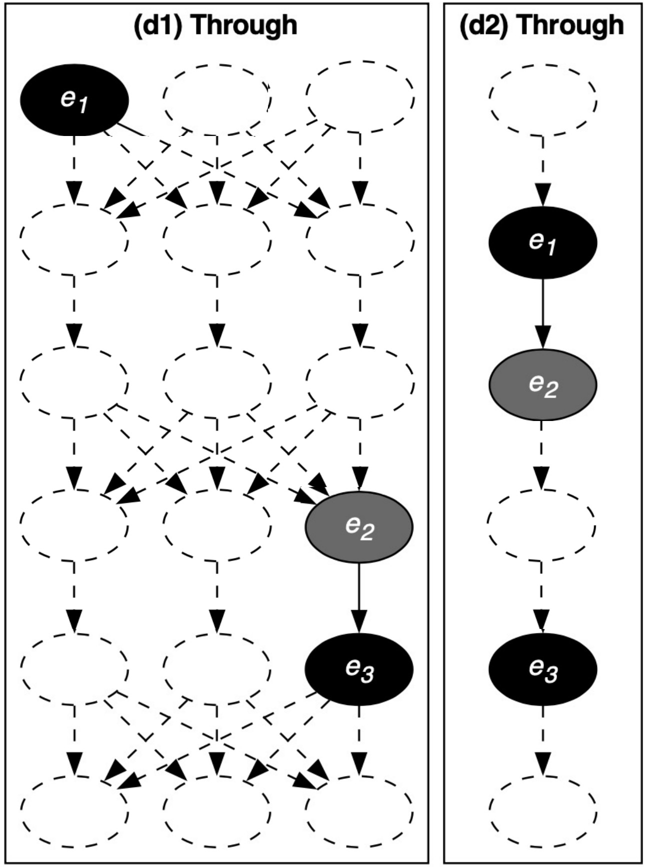
Graph Queries

Reach From



Graph Queries

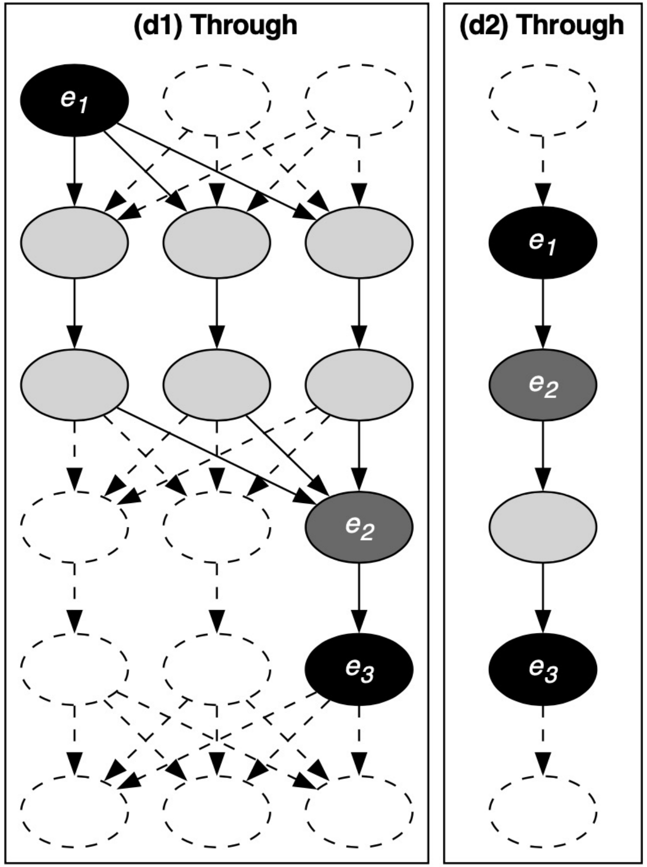
Reach Through



```
function REACHTHROUGH($(\mathcal{G}, v_{origin}, v_{mid}, v_{target})$)
    $\mathcal{G}_{fwd} :=$ REACHFORWARD$(\mathcal{G}, v_{origin})$
    $\mathcal{G}_{back} :=$ REACHBACKWARD$(\mathcal{G}_{fwd}, v_{target})$
    if $v_{mid} \notin$ CUTPOINTS$(\mathcal{G}_{back})$ then
                ▷ Remove midpoint and all edges connected to it
        $\mathcal{G}_{mask} := (V \setminus v_{mid}, \rightarrow_e \setminus (v_{mid}, \_) \cup (\_, v_{mid}))$
        $\mathcal{G}_{path} :=$ PATH$(\mathcal{G}_{mask}, v_{origin}, v_{target})$
    else
        $\mathcal{G}_{path} := (\emptyset, \emptyset)$
```

Graph Queries

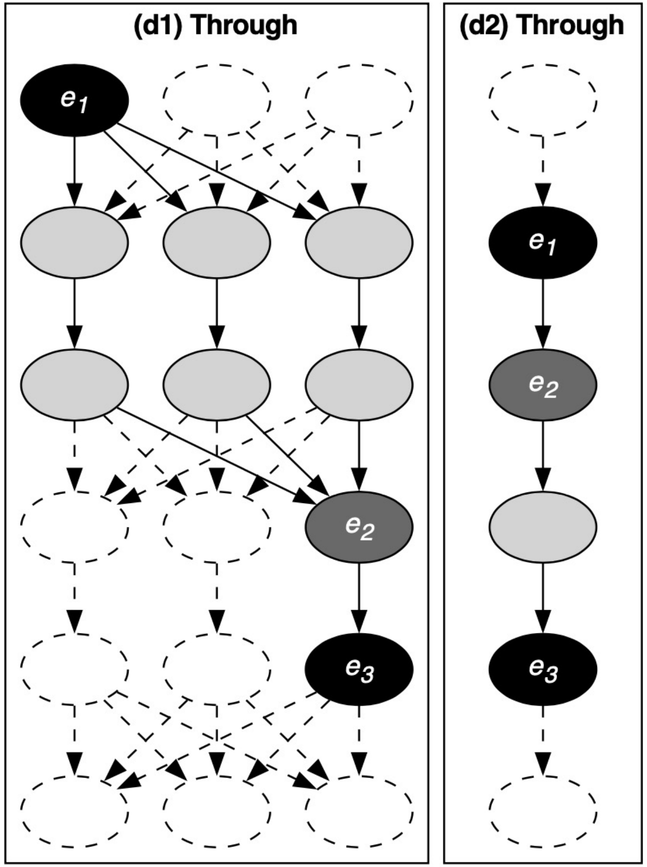
Reach Through



```
function ReachThrough($\mathcal{G}, v_{origin}, v_{mid}, v_{target}$)
    $\mathcal{G}_{fwd} := $ ReachForward($\mathcal{G}, v_{origin}$)
    $\mathcal{G}_{back} := $ ReachBackward($\mathcal{G}_{fwd}, v_{target}$)
    if $v_{mid} \notin$ CutPoints($\mathcal{G}_{back}$) then
            ▷ Remove midpoint and all edges connected to it
        $\mathcal{G}_{mask} := (V \setminus v_{mid}, \rightarrow_e \setminus (v_{mid}, \_\_) \cup (\_\_, v_{mid}))$
        $\mathcal{G}_{path} := $ Path($\mathcal{G}_{mask}, v_{origin}, v_{target}$)
    else
        $\mathcal{G}_{path} := (\emptyset, \emptyset)$
```

Graph Queries

Reach Through



```
function ReachThrough($\mathcal{G}, v_{origin}, v_{mid}, v_{target}$)
    $\mathcal{G}_{fwd} := \text{ReachForward}(\mathcal{G}, v_{origin})$
    $\mathcal{G}_{back} := \text{ReachBackward}(\mathcal{G}_{fwd}, v_{target})$
    if $v_{mid} \notin \text{CutPoints}(\mathcal{G}_{back})$ then
            ▷ Remove midpoint and all edges connected to it
        $\mathcal{G}_{mask} := (V \setminus v_{mid}, \rightarrow_e \setminus (v_{mid}, \_) \cup (\_, v_{mid}))$
        $\mathcal{G}_{path} := \text{Path}(\mathcal{G}_{mask}, v_{origin}, v_{target})$
    else
        $\mathcal{G}_{path} := (\emptyset, \emptyset)$
```

Graph Queries

Others – Validation, Neighbors

Assumption Validation

- Can every error source reach a sink?
- Can every sink be reached from an error source?

Neighbors

- What components, *at a given hierarchical depth*, communicate with a given component?

Agenda

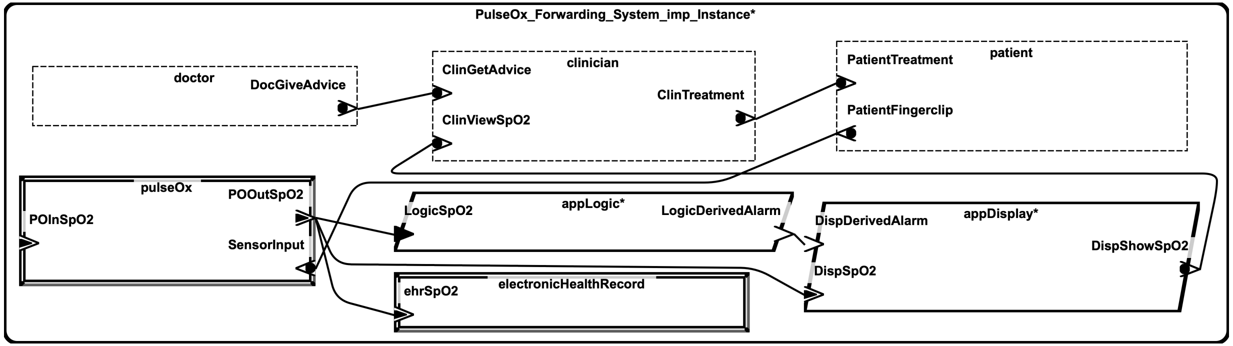
- Introduction & Background
- The OSATE Slicer
- **Evaluation**
  - **Analyses**
  - **Performance**

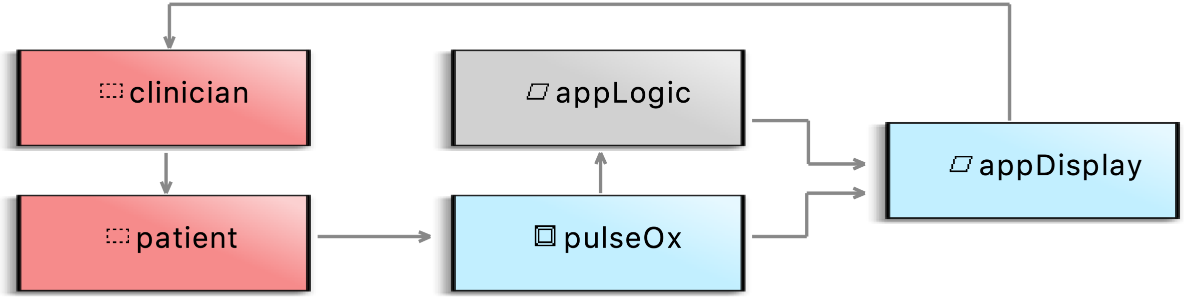
Suitability for Analyses

Safety – Architecture Supported Audit Processor

Questions from Analysis

- Q1: Who can send messages to a component?
- Q2: Who gets messages a component sends?
- A1: “Neighbors” Query



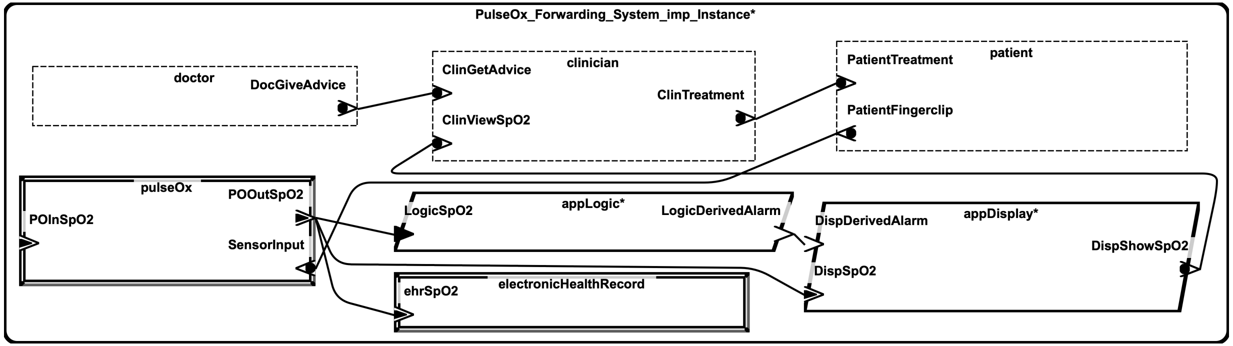


Suitability for Analyses

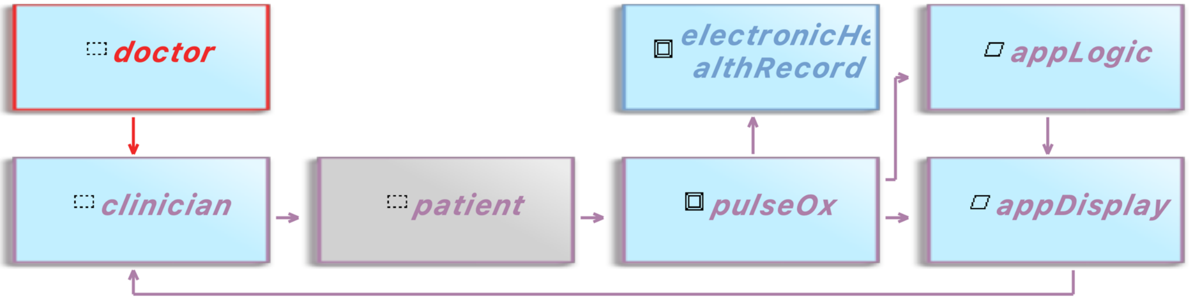
Safety – Architecture Supported Audit Processor

Questions from Analysis

- Q1: Who can send messages to a component?
- Q2: Who gets messages a component sends?
- A1: “Neighbors” Query



- Q3: Who can be affected by affected by a component?
- Q4: Who affects a component?
- Q5: Are there feedback loops present?
- A2: Forward + Backward Slice + Overlap



Suitability for Analyses

Safety, Security, Latency

Safety

Fault Impact

Question: If this error occurs, where does it go? What happens?

Answer: Forward slice

Security

Bell-LaPadula

Classic security policy, 3 of 4 properties can be (potentially) verified using the Slicer.

Attack Trees

Existing implementation is brittle and presents maintenance challenges.

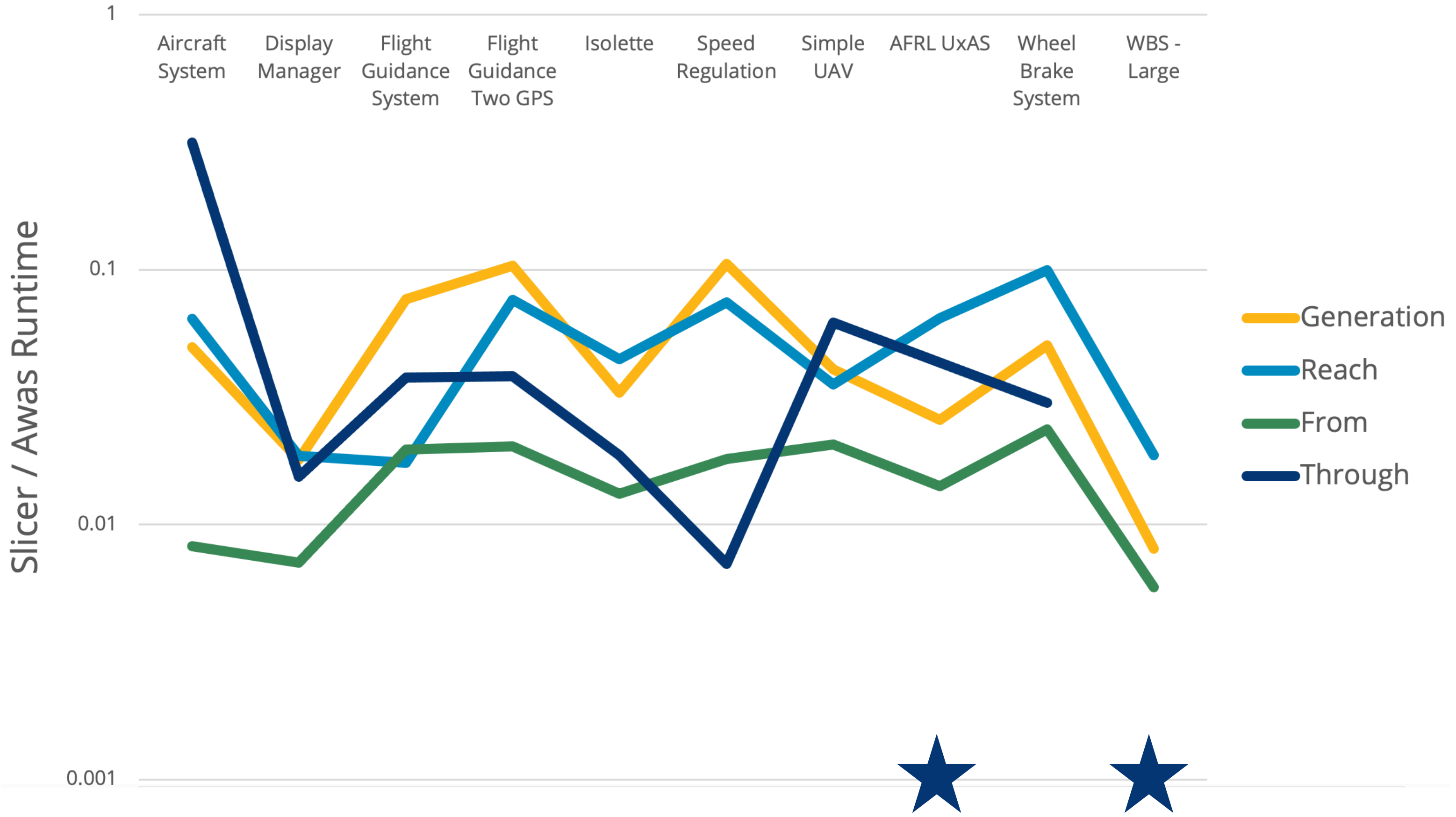
Performance

System Latency

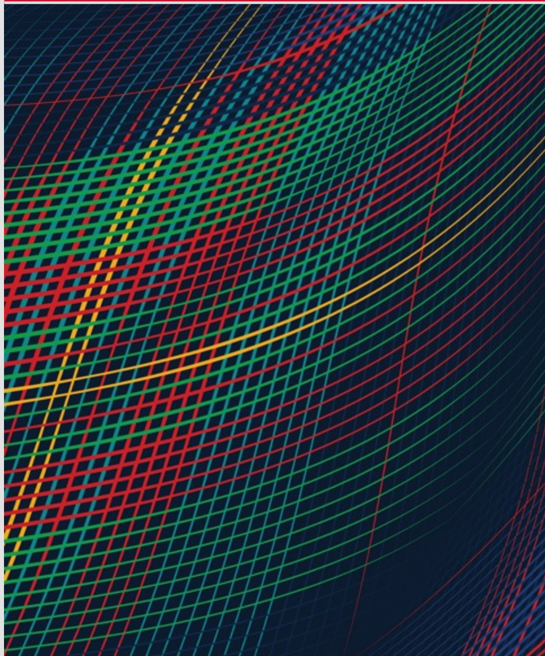
Popular analysis, many special cases. Requires support for additional AADL features, e.g., modes.

Performance

Relative to Awas



Carnegie Mellon University
Software Engineering Institute



The OSATE Slicer: Graph-Based Reachability for Architectural Models

JULY 20, 2023

Sam Procter
sprocter@sei.cmu.edu