Extended Validation and Verification for Situation-Aware Middleware Architectures

SangEun Kim[1], Peter In[2], Ramesh Bharadwaj[3]

[1] Dept. of Computer Science, Texas A&M University, College Station, TX 77840 USA, Tel: 1-979-845-5439, sangeunk@cs.tamu.edu
[2] Dept. of Computer Science, Texas A&M University, College Station, TX 77840 USA, Tel: 1-979-458-1547, hohin@cs.tamu.edu
[3] Center for High Assurance Computer Systems, Naval Research Laboratory, Washington DC 20375 USA, Tel: 1-202-767-7210, ramesh@itd.nrl.navy.mil

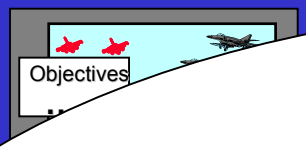
Today's Agenda

- Situation-Aware Middleware Architecture: Introduction
  - Scenario
  - Research Issues
  - Project Goal and Overview
- EVS: A Solution Approach
- Summary

TSCE: A Scenario

JFACC
(Joint Force Air Component Commander)

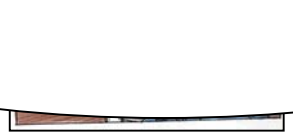
JFC
(Joint Force Cor



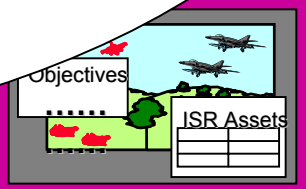
Situation-Awareness

ystems

Target
Systems



ISR



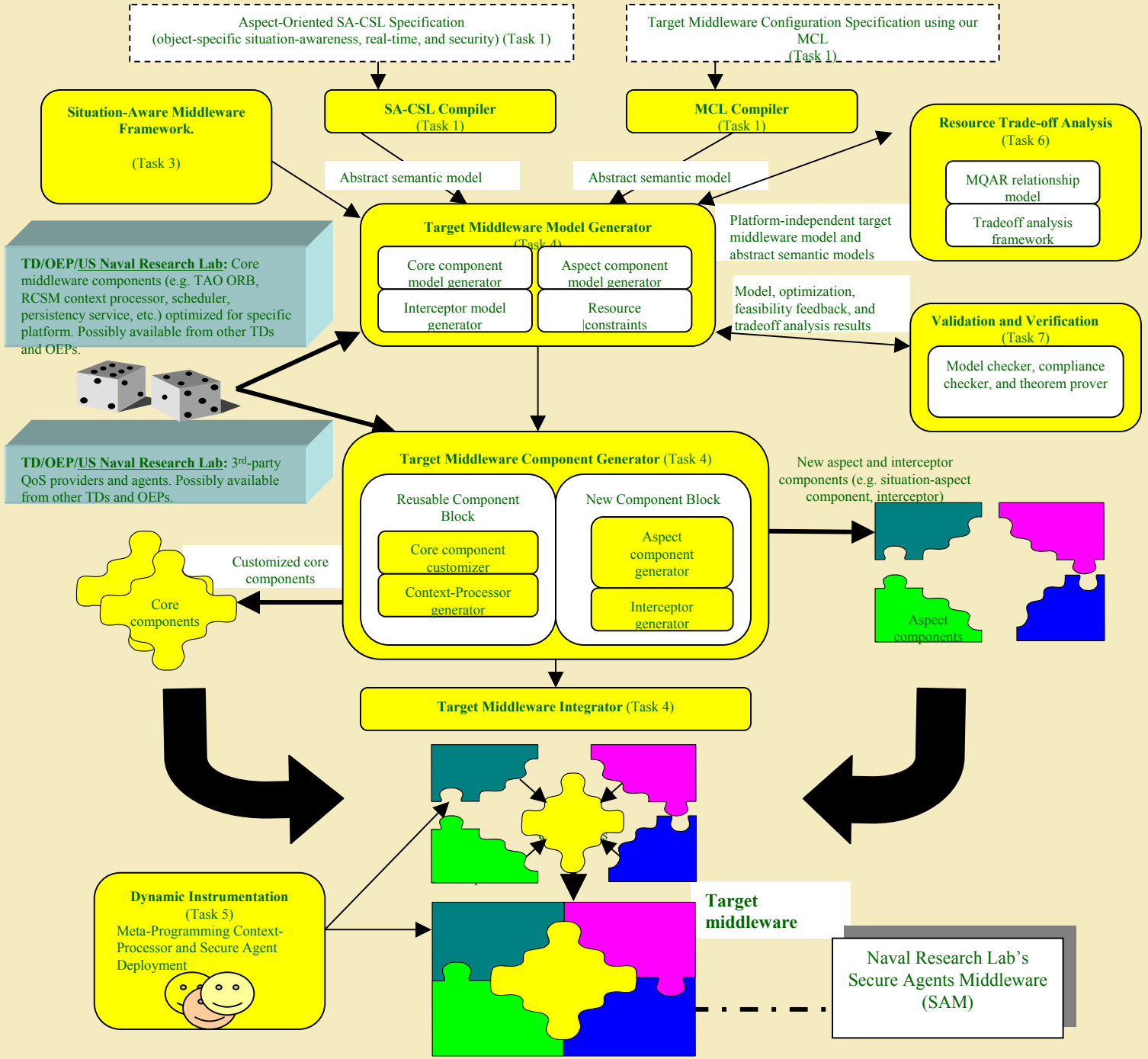
Research Issues

- How to provide timely and transparent support in middleware
  - for application adaptations that are triggered by different situations?
  - for situation-aware, open-standard communication
- How to generate an open middleware framework
  - for generating new and/or reusing 3rd party components?
  - for multiple QoS management mechanism that is tied with various situations of a given mission?
- How to provide efficient, secure services to application developers
  - especially in an multicast and wireless environment in a manner that is survivable and efficient

Project Goals

- Adaptive, Situation-Aware Middleware (SAM) Architectures
  - As the next generation of distributed real-time and embedded (DRE) middleware
  - Adaptable, Secure, Reliable architectures

(Collaboration with Dr. Stephen Yau, Arizona State University)



Overview of Situation-Aware Middleware Architecture

Innovation of SAM

- Situation-awareness.
- Separation of aspects components and middleware core components.
- Automated component integration for combining crosscutting aspects.
- Meta-programmable dynamic instrumentation.
- Trade-off analysis for application and target middleware model optimization.
- Validation and Verification Framework.
- Security and survivability mechanisms utilizing software agents.

V&V: Focus of This Talk

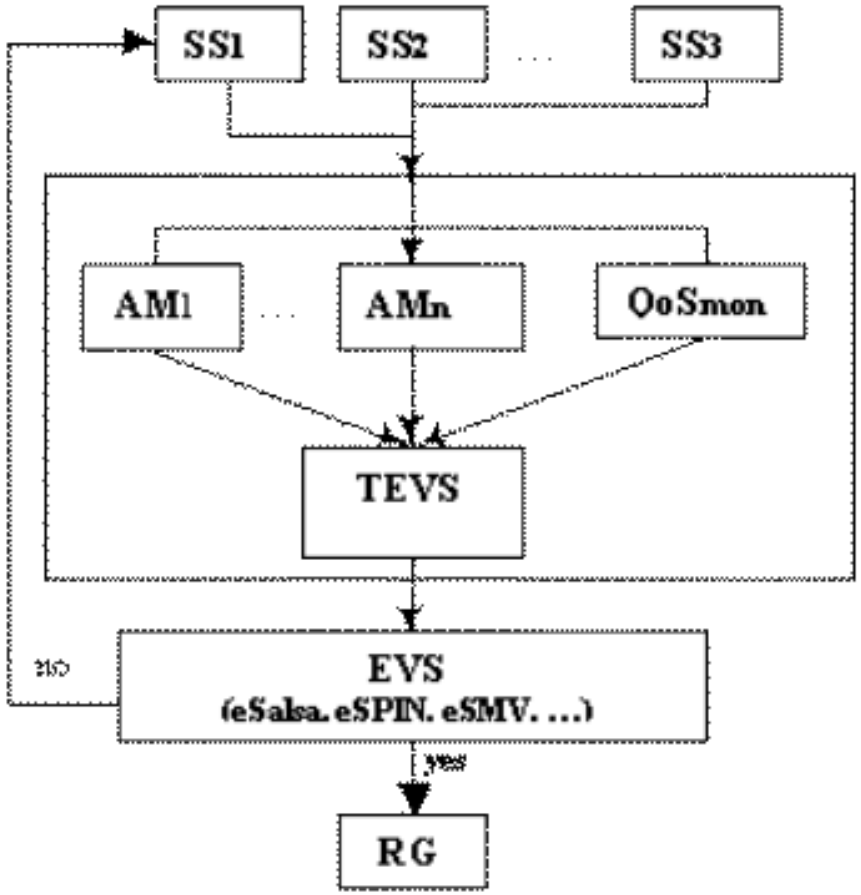
- Difficult to apply traditional V&V technique to situation-awareness applications
  - State explosion problem (huge number of state space)
  - Redundant, unnecessary constraints related to dynamic changing of situations
- Lack of scalability
  - BDD (Binary Decision Diagram)/OBDD (Ordered BDD)
  - Common data types
    - enumerations, integer, real types

Today's Agenda

- Situation-Aware Middleware Architecture: Introduction
- EVS: A Solution Approach
  - Overview
  - Examples
- Summary

EVS: A Solution Approach

- EVS (Extended Validation & Verification System)
  - Combination of model checking and theorem proving (salsa)
  - Automatic property-driven abstraction method



- SS (Situation Specification)
- AM (Abstraction Mechanism)
- QoSmon (QoS monitor)
- TEVS (Translator for EVS)
- EVS (Extended Validation and Verification System)
- RG (Report Generator)

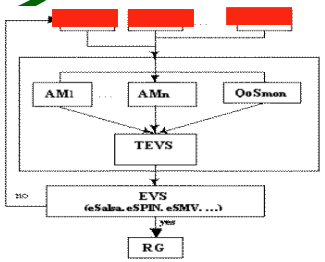
Predator: An Example

- Total Ship Computing Environment (TSCE)

Predator's mission is to take reconnaissance pictures and send back the pictures to the carrier.

Predator command and control in the carrier.

Step 1. Situation Specification



Mission 1: Destroy an enemy target.

Resources:

missile, radar, fuel, etc.

Actions: launch missile(), guide missile()

QoS:

1) The missile should be launched within **n** seconds after the command is received from the carrier.

Situations:

Situation 2: If it receives a "destroy" command, the drone should launch missile.

Situation 3: After the missile is launched and before it hits the target, the radar system should guide the missile.

Mission 2: Reconnaissance

Resources:

radar, communication system, fuel, etc.

Actions: scan(), send-information()

QoS:

1) Each scan action has to be completed by **m** seconds.

2) The information sent back to the carrier should not be tampered.

Situations:

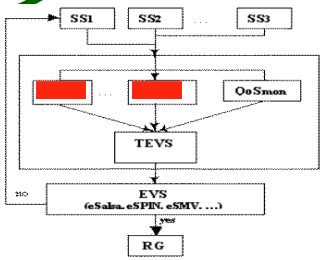
Situation 1: If the drone is in enemy territory, then every **k** seconds (k>m), the radar should perform a scan action and a send-information action.

Situation Specification (Continued)

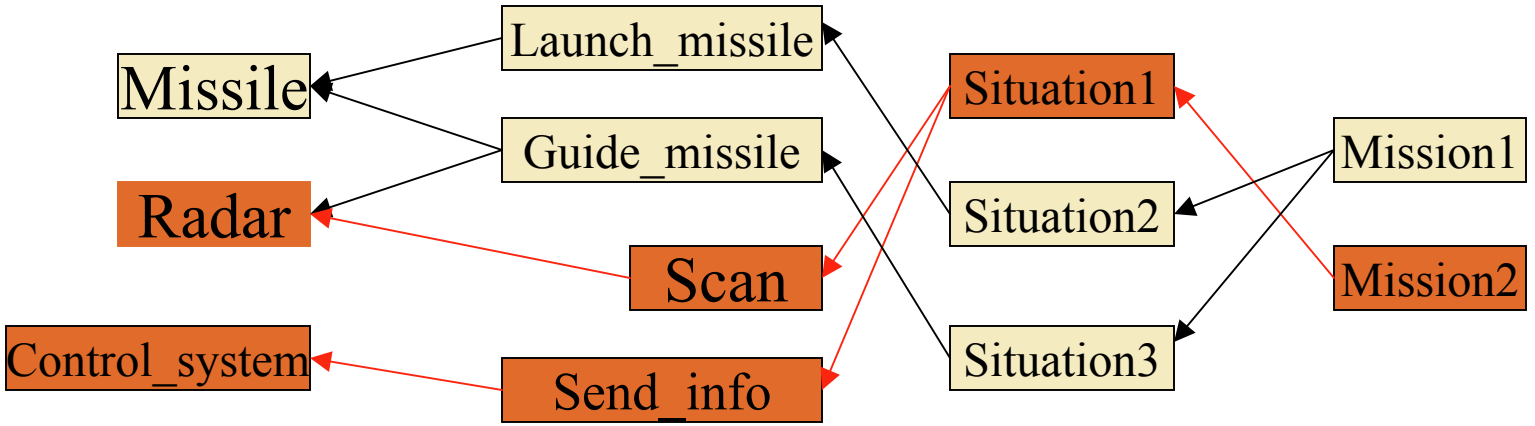
```
QoS-Security {
        Entity goal;Action in;
        Action out;Mechanism m1;
        out.input=m1(in.result);
} Sec1;
QoS-RealTime {
        Int Duration;
        Int Importance;
} RTC1;
RTScan = new RTC1 (m, 0);
RTLaunchMissile = new RTC1 (n, 1);
RTGuideMissile = new RTC1 (null, 1);
SecureSendInfo = new Sec1 (Carrier, scan, sendInfo, PublicEncryption);
Resource {
        Int Missile; Int Communication;
        Int Radar; Int[] getResourceAvailable();
} DroneResource;
ResrScan = new DroneResource (0, 0, 1);
ResrSendInfo = new DroneResource (0, 1, 0);
ResrLaunchMissile = new DroneResource (1, 0, 0);
ResrGuideMissile = new DroneResource (1, 0, 1);
```

```
Situation-aware-object {
    Situation1: Location is in enemy territory, every k seconds Situation1 is true;
    Situation2: Drone receives "destroy" command, and missile has not been launched
        yet;
    Situation3: Missile has been launched and it has not hit the target yet.
[local] [Activate at Situation1] scan ()
        RequireResources ResrScan
        withQoSConstraint RTScan;
[outgoing] [Activate at Situation1] sendInfo ()
        RequireResources ResrSendInfo;
        withQoSConstraint SecureSendInfo;
[local] [Activate at Situation 2] launchMissile ()
        RequireResources ResrlaunchMissile
        WithQoSConstraint RTSLaunchMissile;
[outgoing] [Activate at Situation3] guideMissile ()
        RequireResources ResrGuideMissile
        WithQoSConstraint1 RTGuideMissile;
        WithQoSConstraint2 … … another securityQoS
} DroneControl;
QoSExceptionHandler {
        fail RTScan do action1;
        fail SecureSendInfo do action2;
        … …
}DroneExceptionHandler;
```

Step 2. Abstract Mechanism



- AM1: Remove irrelevant information
  - Based on analysis of relationship between variables



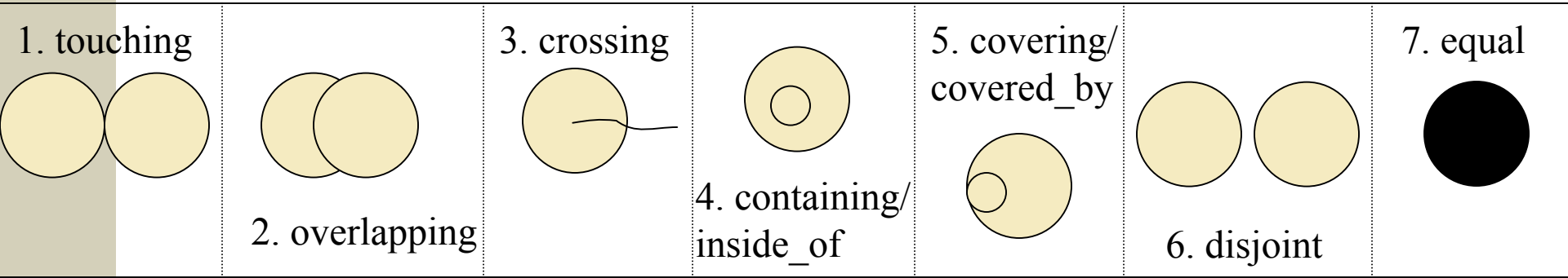
Dependency graph

Abstract Mechanism (Continued)

- AM2: Spatial Information Reduction
  - Based on spatial analysis based on spatial relationships

$L_{BattleField}$ = {zone1, zone2, zone3};
L_{Enemy} = {zone1, zone2};
Loc = {L_{Enemy}, $L_{BattleField}$};
L_1 Loc; L_2 Loc;

scan (L_1==L_{Enemy} AND L_2 == $L_{BattleField}$);
→ scan (L_1 == L_{Enemy});

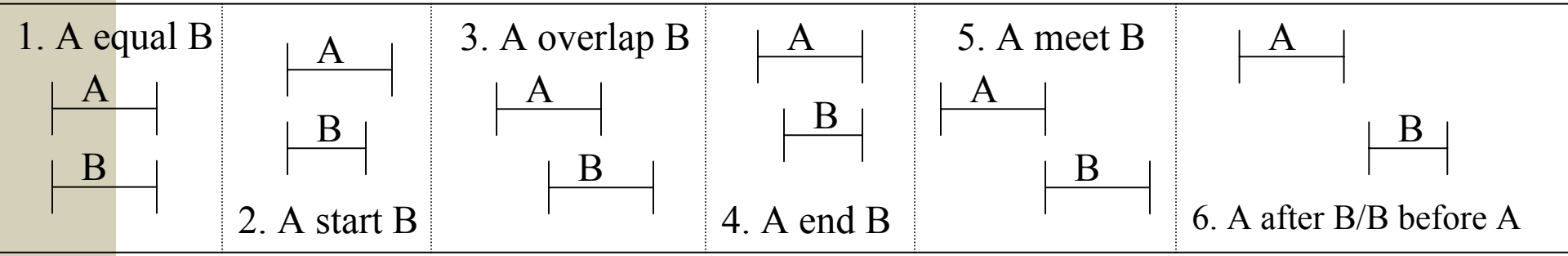


Abstract Mechanism (Continued)

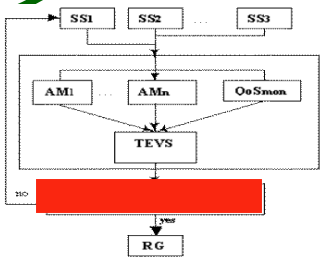
- AM3: Temporal Information Reduction
  - Based on temporal analysis based on temporal relationship

scan(); AFTER launchMissile();
launchMissile(); AFTER guideMissile();

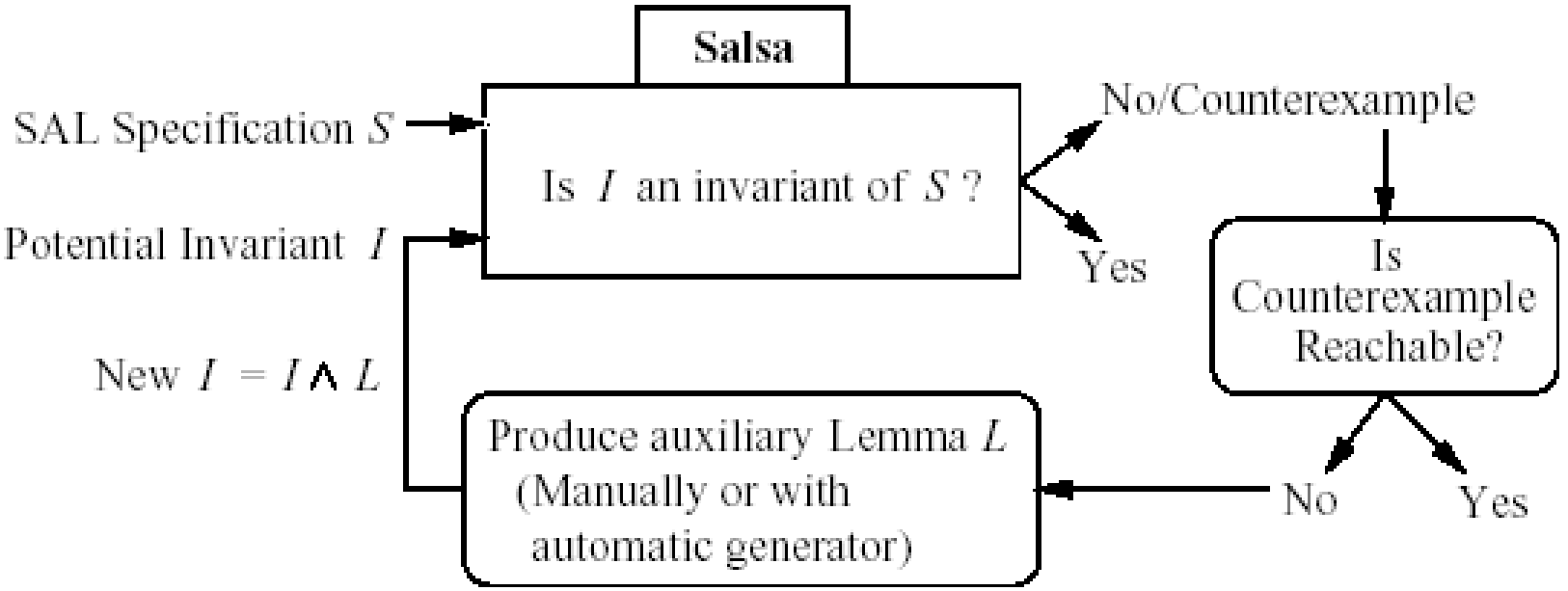
→ scan(); AFTER guideMissile()



Step 3. EVS



- Situation-aware Salsa
  - Invariant checker for situation-aware specifications



<Process for applying Salsa>

Salsa: Specification

```
module TSCE_drone

type definitions
        OnOff : {On, Off};

monitored variables
        Missile, Radar, Control_System : OnOff;

controlled variables
        TSCE_drone : OnOff;

internal variables
        launchMissile, guideMissile, scan, sendInfo : bool;
        Situation1, Situation2, Situation3 : bool;
        Mission1, Mission2 : bool;

guarantees
        /* true properties */
        Property1 = @T(Radar = On) when (Situation1) => scan';
        Property2 = (Missile = On and Radar = On) => guideMissile;
        /* false properties */
        Property3 = (Missile = On and launchMissile) =>  not scan;
        Property4 = (Radar = On and guideMissile) => Missile = Off;

definitions
  var launchMissile initially false :=
        ev
        [] @F(scan) -> false
        [] @T(scan) when (Missile = On) -> true
        [] @T(guideMissile) when (not scan) -> false
        ve

  var guideMissile initially false :=
    ev
      [] @F(scan) -> false
      [] @T(scan) when (Missile = Off or Radar = Off) -> false
      [] @T(scan) when (Missile = On and Radar = On and launchMissile)  -
         > true
    ve

  var scan initially false :=
        ev
                [] @T(Radar = On) when (Situation1) -> true
                [] @T(Radar = On) when (not Situation1) -> false
        ve
end module
```

Salsa: The Result

Analyzing SAL specification in file: tcse.sal.
Checking disjointness of all modules.
Checking module TSCE_drone
Number of Nontrivial Atoms: 0
Checking launchMissile ... disjoint.
Checking guideMissile ... disjoint.
Checking scan ... disjoint.
All checks passed.
Number of failed/passed verification conditions: 0/7
Time (total) : 0.226
Rewriting : 0.078
Partitioning : 0.000
Integer solving : 0.000
Bdd ops(total,gc) : 0.058, 0.000

BDD statistics.
Number of variables : 25
Number of nodes
User : 96
Total : 467
Table size : 65536

Checking coverage of all modules.
Checking module TSCE_drone
Number of Nontrivial Atoms: 0
All checks passed.
Number of failed/passed verification conditions: 0/0
Time (total) : 0.076
Rewriting : 0.013
Partitioning : 0.000
Integer solving : 0.000
Bdd ops(total,gc) : 0.000, 0.000

BDD statistics.
Number of variables : 25
Number of nodes
User : 1
Total : 2
Table size : 65536

Checking guarantees in all modules.
Checking module TSCE_drone
Number of Nontrivial Atoms: 0
Checking Property1 ... pass
Checking Property2 ... fail
Checking Property3 ... fail
Checking Property4 ... fail

Checks failed for: Property4, Property3, Property2
Number of failed/passed verification conditions: 3/1
Time (total) : 0.315
Rewriting : 0.131
Partitioning : 0.000
Integer solving : 0.000
Bdd ops(total,gc) : 0.072, 0.000

BDD statistics.
Number of variables : 25
Number of nodes
User : 119
Total : 528
Table size : 65536

Salsa: Extension to Situation-Aware

Extension for Spatial Relationship

definitions

…….

```
var TSCE_drone=
        case Mission1
          [] @T(launchMissile) CROSSING @T(enemy_area) ->
            if []true -> true []false -> false fi
        esac
        case Mission2
          [] @T(scan) -> if []true -> true []false -> false fi
        esac
```

Extension for Temporal Relationship

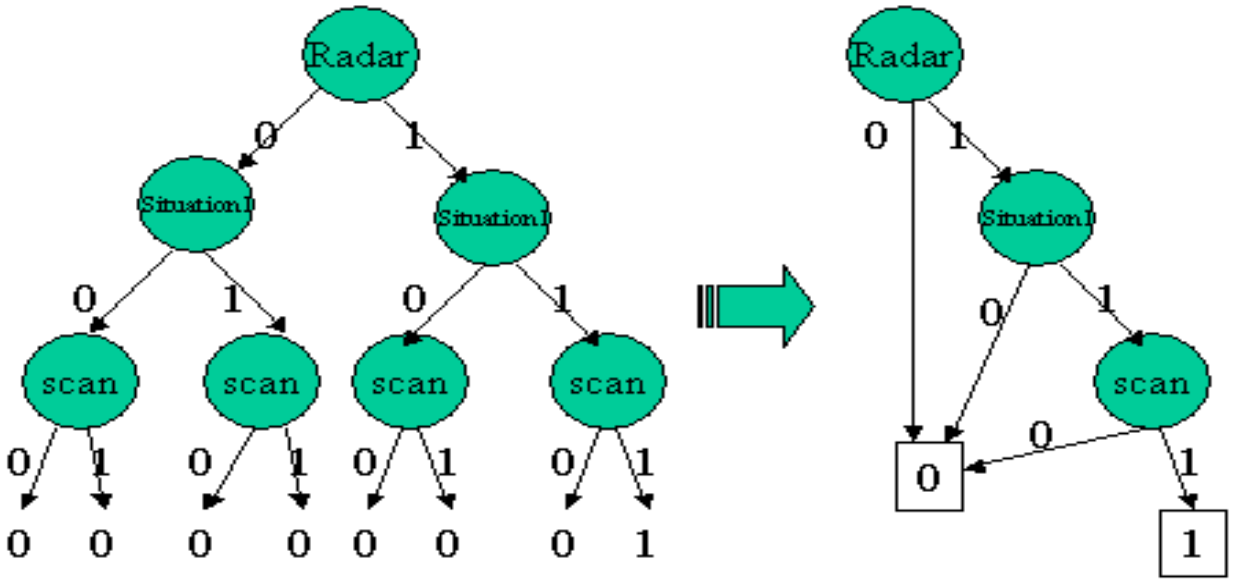
definitions

…….

```
var TSCE_drone=
        case Mission1
          [] @T(launchMissile) BEFORE @T(guideMissile) ->
            if []true -> true []false -> false fi
        esac
        case Mission2
          [] @T(scan) -> if []true -> true []false -> false fi
        esac
```

EVS: Extension to OBDD

- BDD(Binary Decision Diagram) and OBDD(Ordered BDD) for property1
  - (Radar = On AND Situation1) => scan;



<BDD> <OBDD>

EVS: CTL Capability

- Check a CTL formula,
  - AG(scan -> AF guideMissile)

<Step1>

AG(scan -> AF guideMissile) ≡ ~EF(scan ^ ~guideMissile)

<Step2>

S(scan) = {1}

S(~guideMissile) = {1,2,3}

S(EG ~guideMissile) = {1,2,3,5}

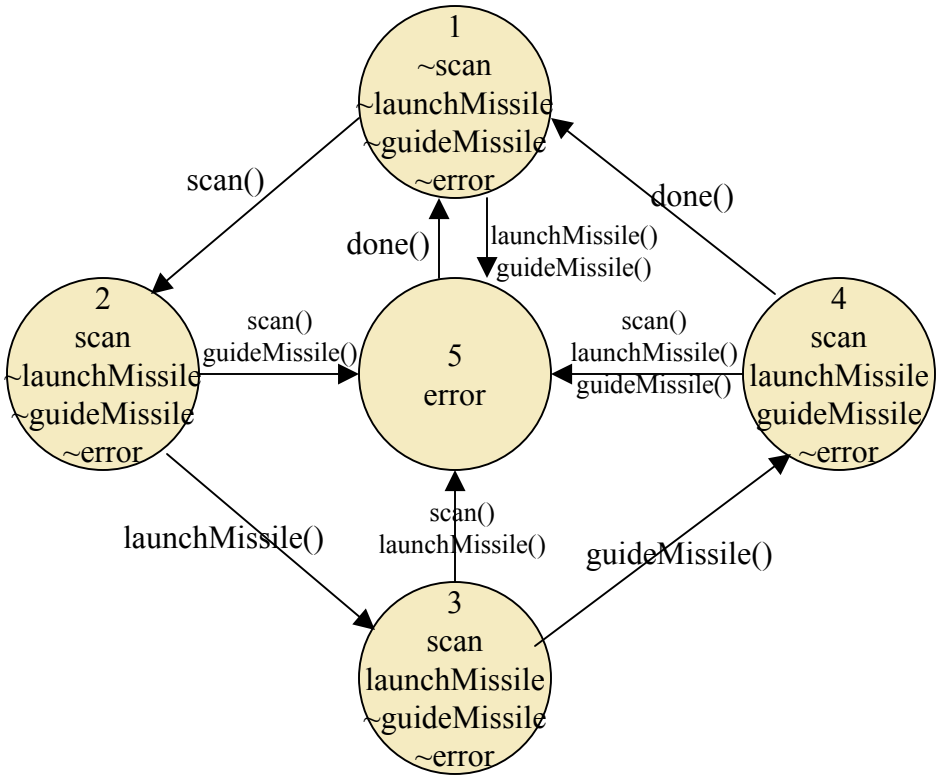
<Step3>

S(scan ^ EG ~guideMissile) = {1}

S(EF(scan ^ EG ~guideMissile)) = {1,2,3,4,5}

<Step4>

S(~EF(scan ^ EG ~guideMissile)) = Ø



<Kripke Structure for TSCE_drone>

Summary

- Extended V&V for Situation-Aware Middleware Architectures
  - Redundant, unnecessary constraints related to dynamic changing of situations
    - Represent by Situation Specification
    - Reduce by Situation-aware Abstract Mechanisms (Spatial and Temporal).
  - Reduce the number of state space for V&V
    - By salsa (combining model checking and theorem proving)

Contact Points

- Dr. Peter In

Assistant Professor
Computer Science Department
Texas A&M University
College Station, Texas 77843-3112
Voice: +1-979-458-1547
Fax: +1-979-847-8578
Email: hohin@cs.tamu.edu
Web: http://www.cs.tamu.edu/faculty/hohin

- Dr. Ramesh Bharadwaj

Center for High Assurance Computer Systems
Naval Research Laboratory
Washington DC 20375 USA
Email: ramesh@itd.nrl.navy.mil
Phone: +1-202-767-7210
Fax: +1-202-404-7942