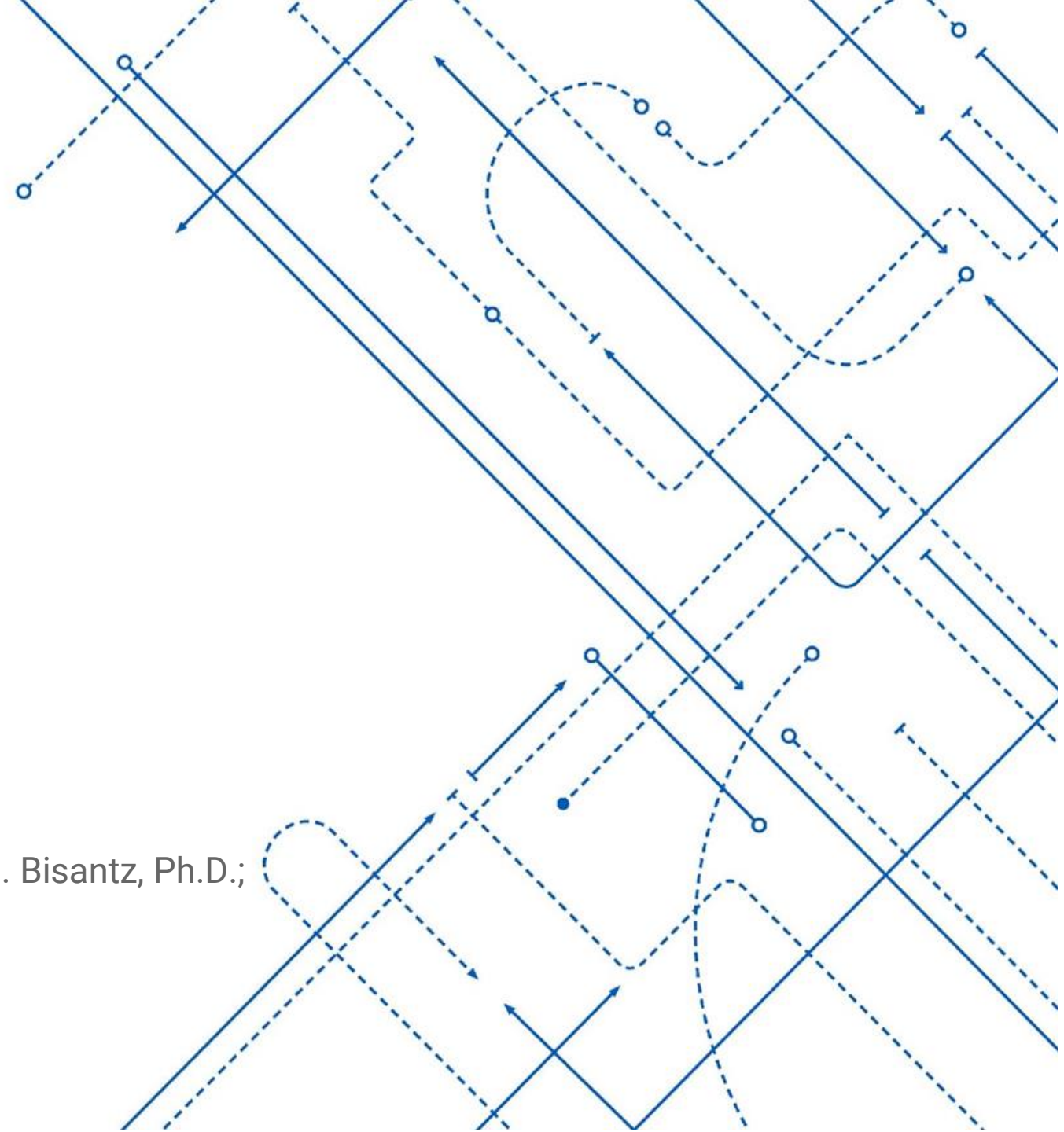
Mental Models for Cybersecurity

A Formal Methods Approach

Adam M. Houser
May 25, 2018

Committee: Matthew L. Bolton, Ph.D. (Chair); Ann M. Bisantz, Ph.D.;
Jun Zhuang, Ph.D.

University at Buffalo The State University of New York



Agenda

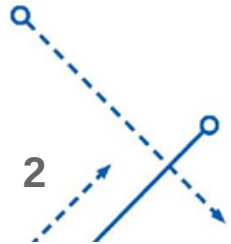
Motivation and brief backgrounders on relevant topics

Overview of the method

Use case 1: Configuration errors in Amazon cloud storage

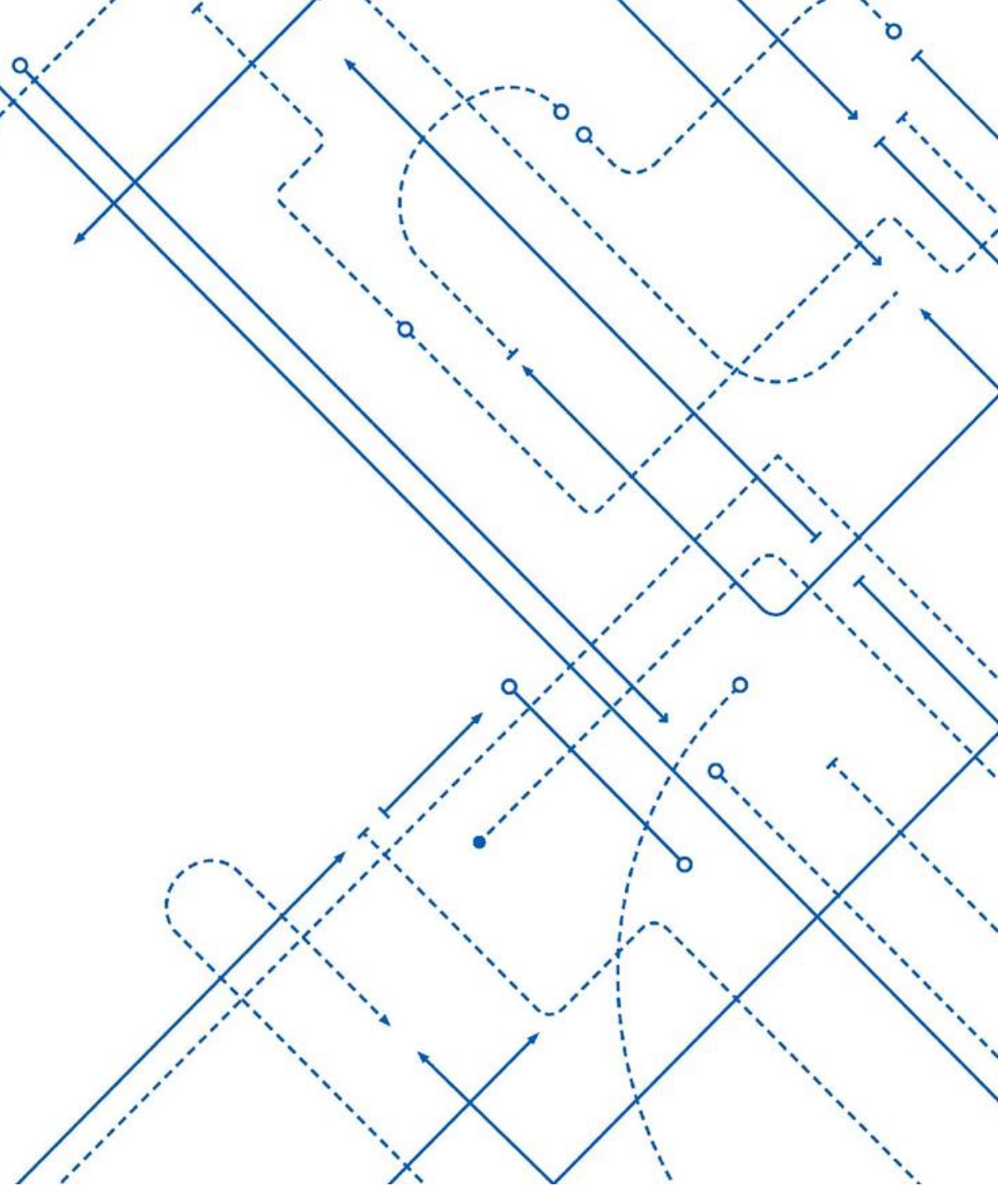
Use case 2: User actions when receiving phishing emails

Discussion and targets for future work



University at Buffalo The State University of New York

Motivating the Dissertation



Why formal methods in cybersecurity?

As a domain area, cybersecurity has become a critical global concern

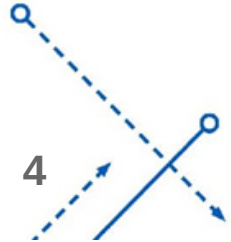
- Nation-state cyberattacks, ransomware, global DDoS, crime (FBI: Cyber Crime, 2016)
- Threat actors turning to human exploits

One promising approach: explore cyber through mental models and HAI

- Address not only “what” humans do, but “why” as well
- Understand how these misunderstandings can lead to vulnerabilities, breaches, system failure

Formal methods offer techniques for discovering these events

- Very good at discovery of rare, unanticipated conditions (exhaustive search, counterexample traces)
- Useful, illuminating synergistic intersection of these topics



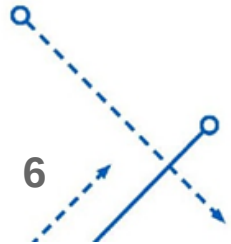
Background: Cybersecurity

- Security: $75 billion industry in 2015, projected $170 billion by 2020 (Gartner, 2015; Morgan, 2016)
- Crime: $400 billion in theft and loss in 2015, projected $2 to $3 trillion by 2020 – 2021 (Juniper Research, 2015)
- Security investments have made “hacking” more difficult (Collet, 2014)
  - Gone are the days of Hollywood-style exploits
  - Mr. Robot: accurate but uncommon
- Humans as critical linchpins in the cyber kill chain
  - Phishing for credential theft
  - Malware for remote access
  - Ransomware for extortion

Background: Mental models

- To bolster end-user cyber defenses: training
  - Mixed empirical data on phishing clickthrough and recidivism (repeat offenders) (Kumaraguru, et al., 2009; Sheng, et al., 2010; Siadati, et al., 2017)

- Mental models may help explore user-based exploits more deeply
  - A “small-scale” mental representation of some aspect of the world (Craik, 1943)
  - Describes functionality of the target system (Norman, 1983, 2013)
  - Often incomplete, unscientific, can harbor “superstitious beliefs” about the system, but not “wrong”
  - Runnable forwards and backwards (de Kleer and Brown, 1981; Norman, 1983)

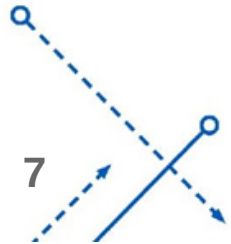


Background: Folk models

- What are folk models?
  - Phenomenological models shared among similar members of a culture (D'Andre, 1987)
  - Useful, but potentially immune to falsification, posit without measurement (Dekker and Hollnagel, 2004)

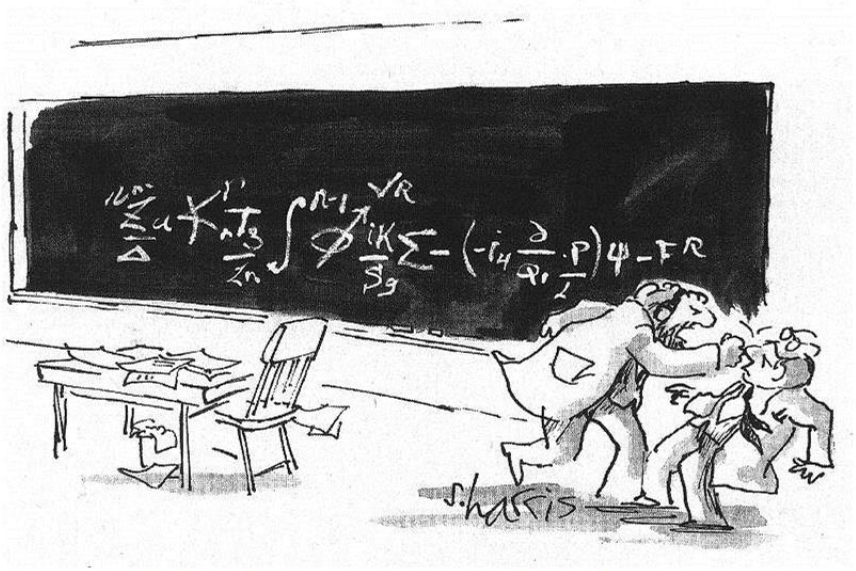
- Existing work in folk modeling of user computer threats, user responses, security devices (Wash, 2010; Kauer, 2013)
  - Heavily reliant on metaphor, lack runnability (Camp, 2004, 2009)
  - Some effort to add rigor (Blythe and Camp, 2012)

- Folk models are still useful, warts and all
  - Contain valuable insight on user security responses
  - High face validity of response data

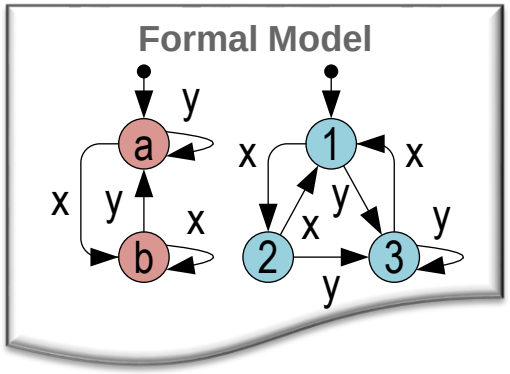


Background: Formal methods

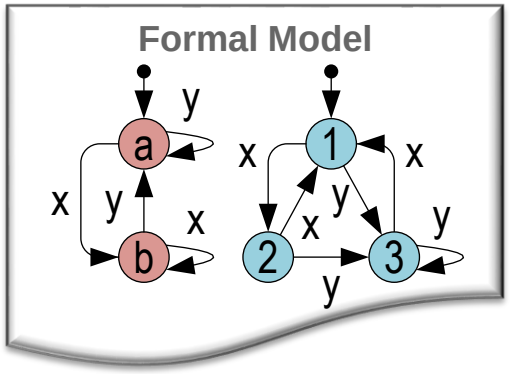
- Formal methods: mathematical languages and techniques for modeling, specifying, and verifying system models and their properties (Baier and Katoen, 2008; Bolton, 2013)
- Model checking: highly-automated approach to formal verification
- Mathematical proof that condition does or does not exist (exhaustive search, counterexamples)



Background: Model checking



Background: Model checking

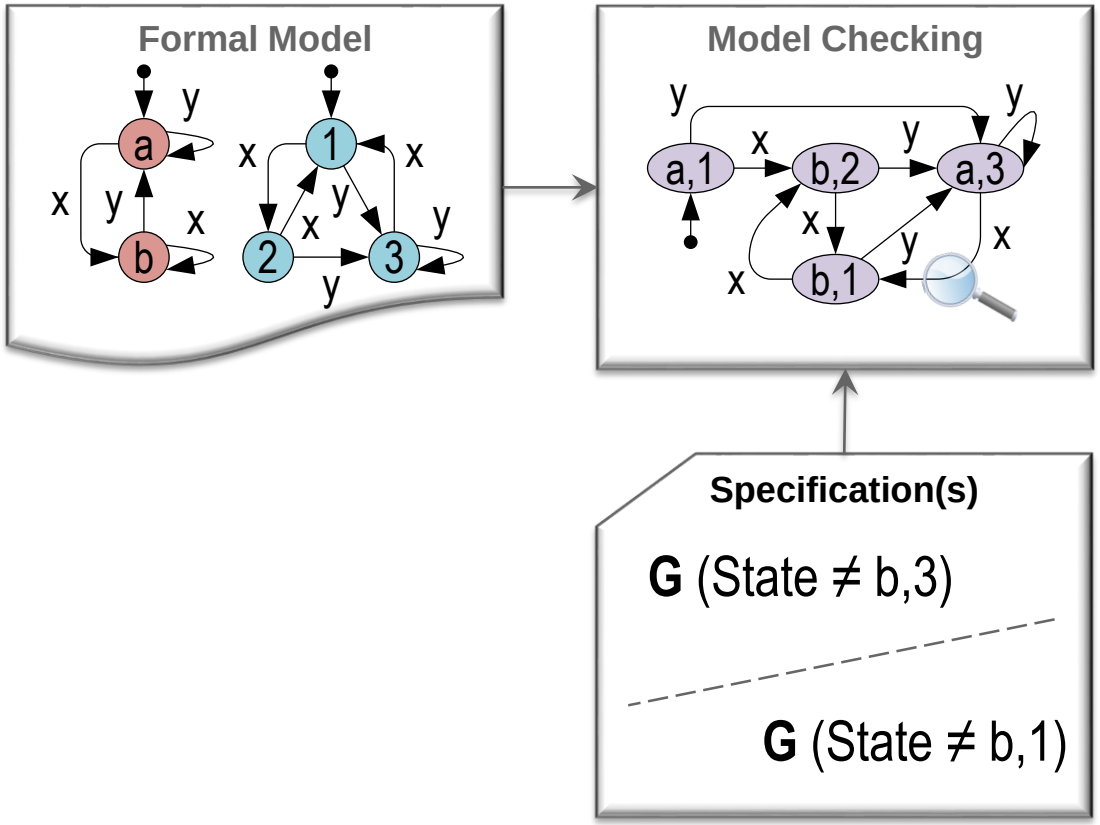


Specification(s)

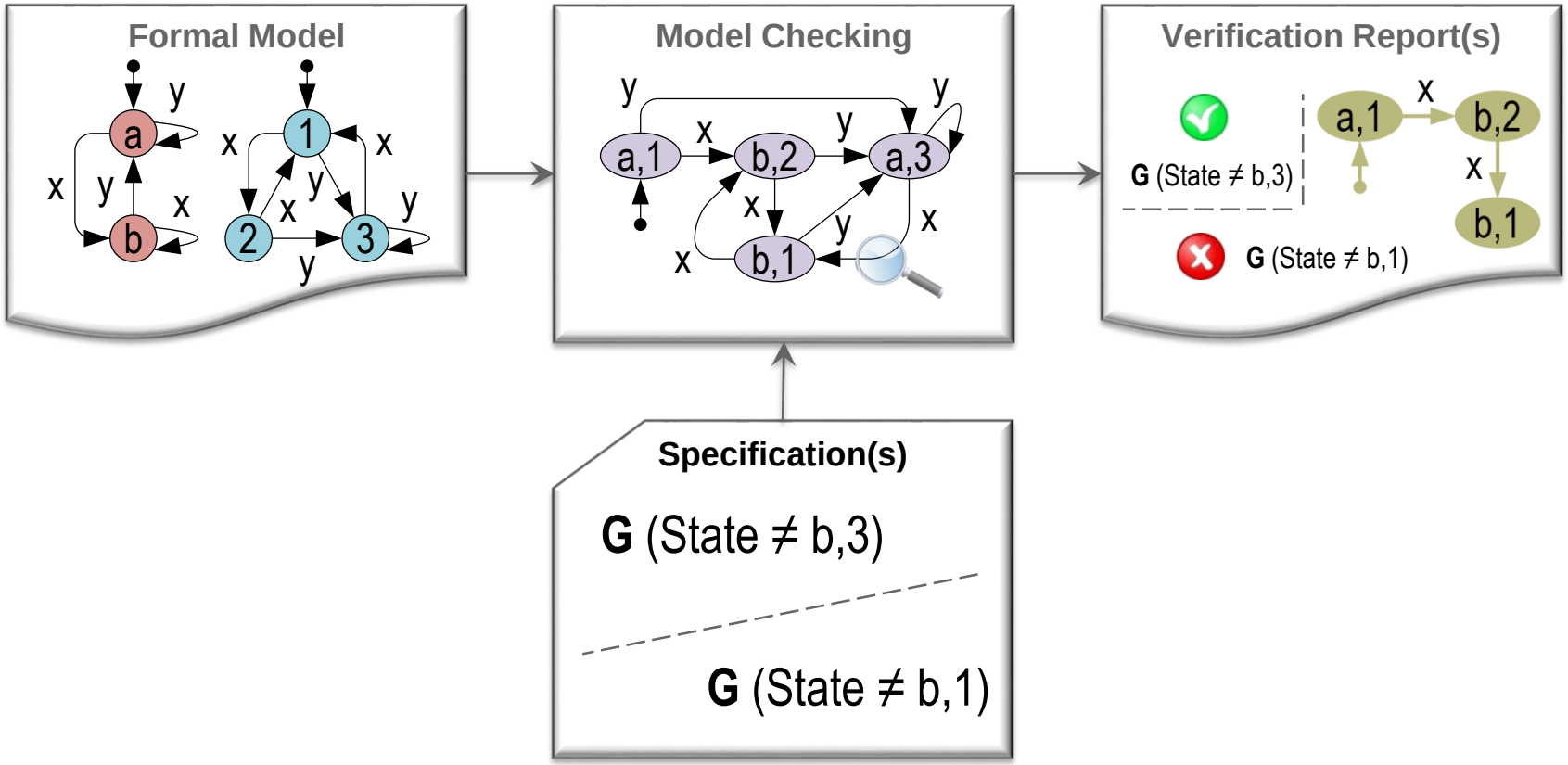
G (State ≠ b,3)

G (State ≠ b,1)

Background: Model checking



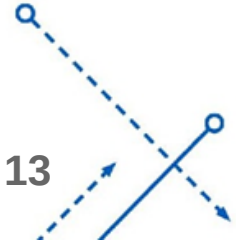
Background: Model checking



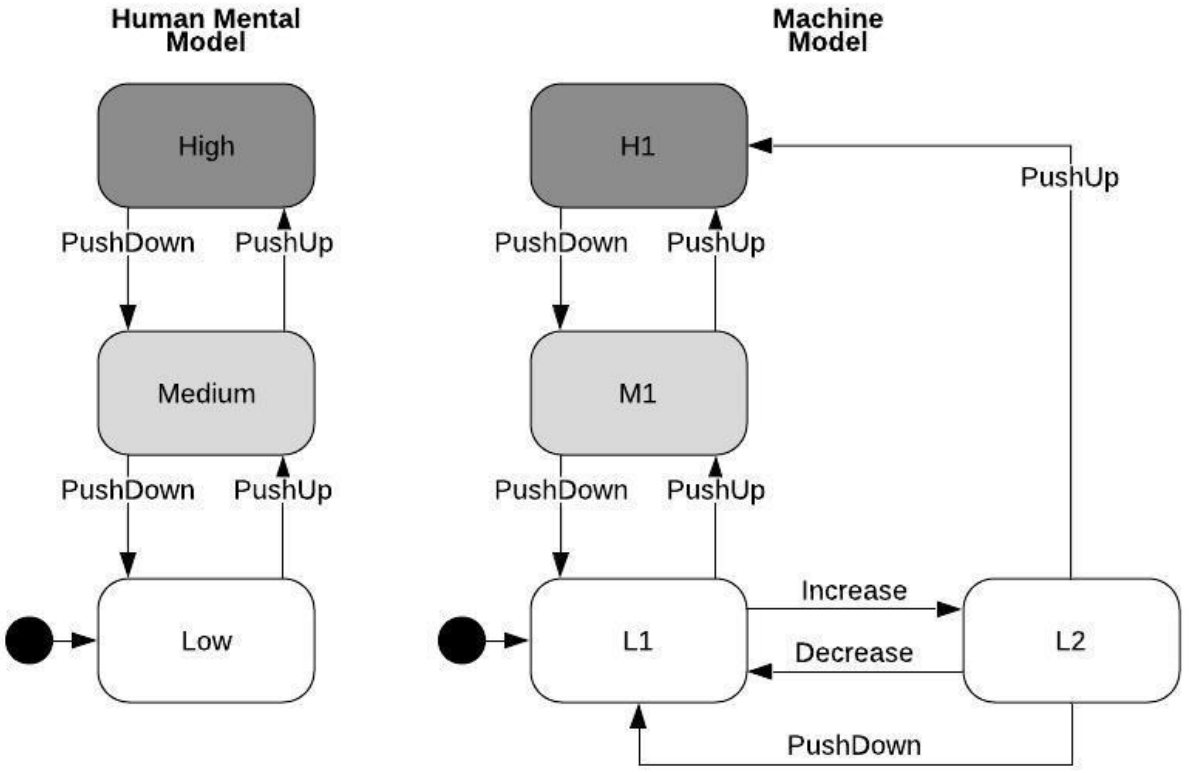
Background: Formal methods

- Mode confusion: mismatch between user mental model of system state and state of actual system
  - Cruise-control systems (Degani, Shafto, and Kirlik, 1999)
  - Aircraft autopilot (Degani and Heymann, 2002)
  - Aircraft auto-land (Oishi et al., 2003)

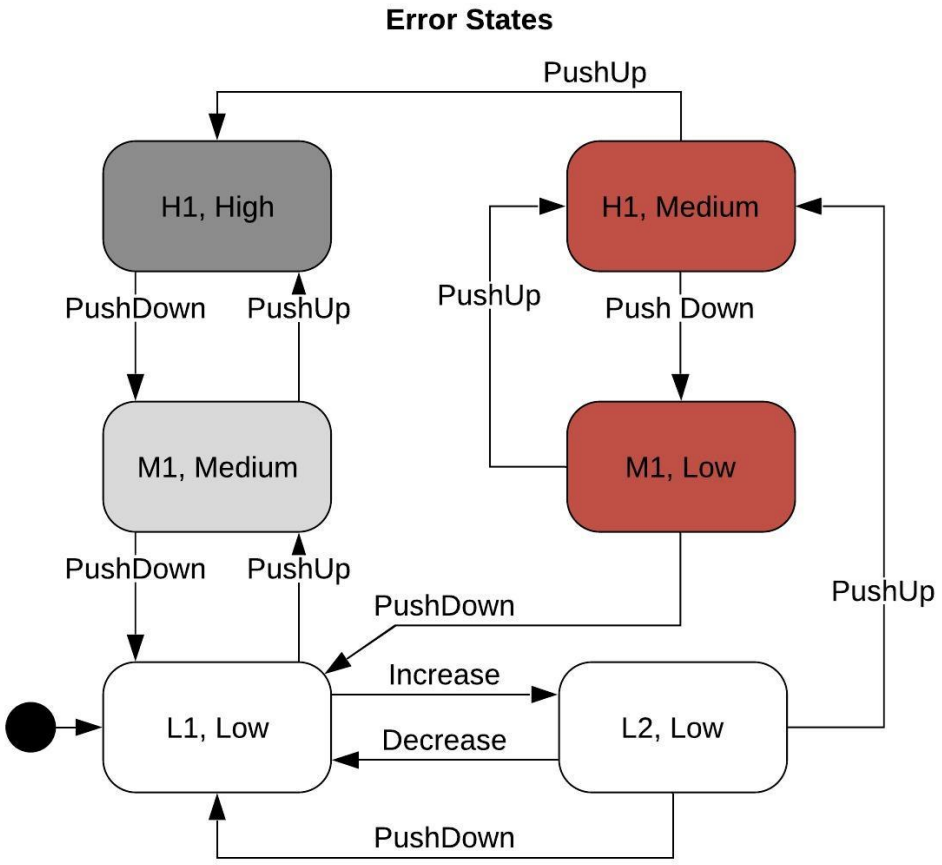
- Error and blocking states (Degani and Heymann, 2002)
  - States should transition synchronously
  - Error state: user thinks move is legal, machine state is illegal
  - Blocking state: system transition happens or is possible, but user doesn't see it or is unaware of it



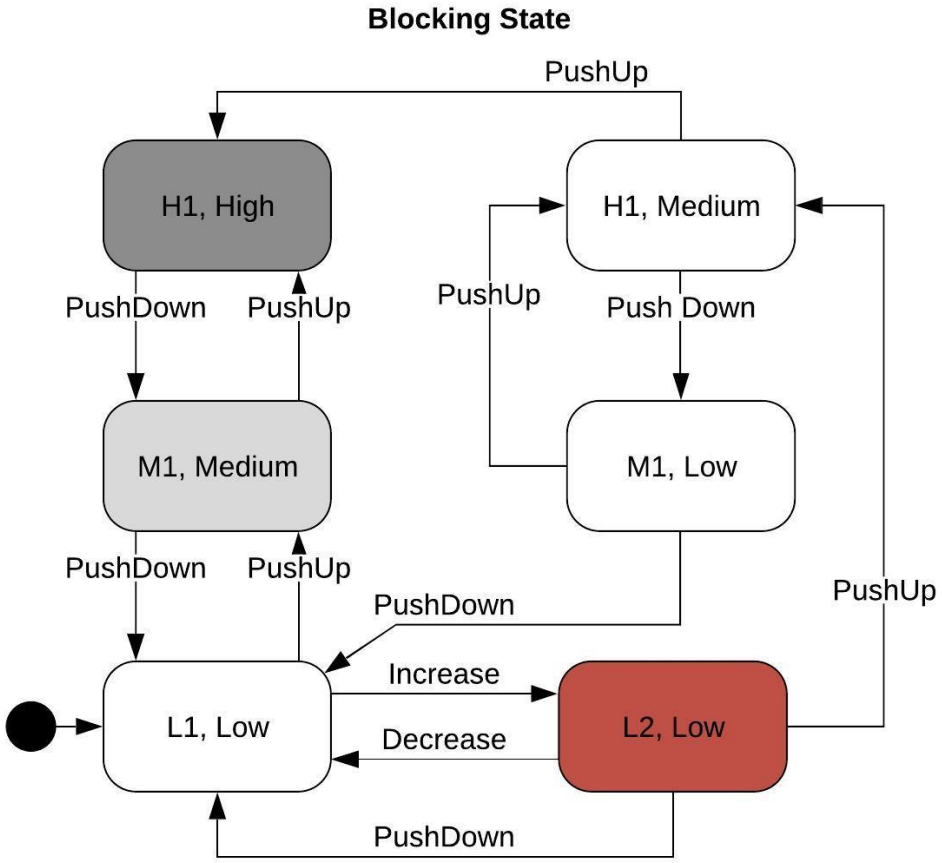
Degani: human and machine models



Degani's error states



Degani's blocking state



University at Buffalo The State University of New York

Research Questions

Research Question 1

How can rigorous characterization of **computer user mental models** and their treatment of **cyber threats** help network defenders become more resilient to cyberattacks that exploit human users?

Research Question 2

How can **formal methods techniques** be extended to give analysts the ability to **discover and describe real-world security vulnerabilities** that arise from the interaction between user mental models of computer security tools and the tools themselves?

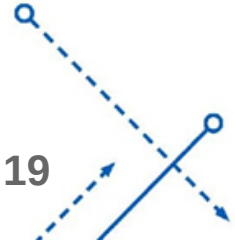
Research Objectives

Establish a rigorous framework for exploring user mental models of computer security risks with formal methods techniques.

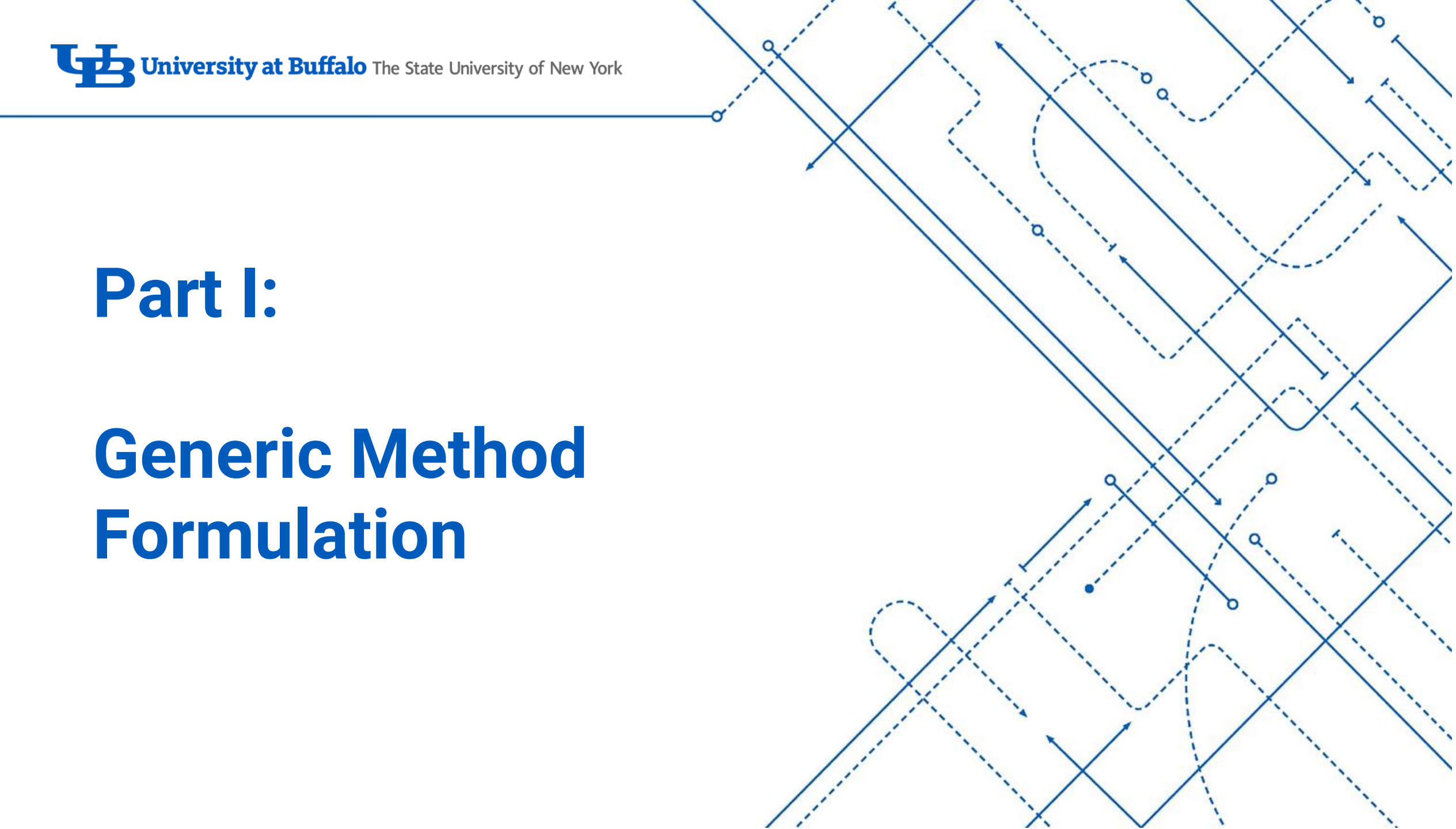
Demonstrate that these risks correspond to vulnerabilities that attackers could potentially exploit in modern computer systems.

Extend our framework to incorporate folk models of cybersecurity risks.

Analyze data collected from multiple sources to validate our modeling approach by discovering potential configuration errors anticipated with our method



University at Buffalo The State University of New York



Part I:

Generic Method Formulation

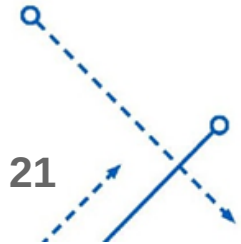
Problem overview

Motivation: user-enabled attacks are a major cybersecurity problem, expected continual growth

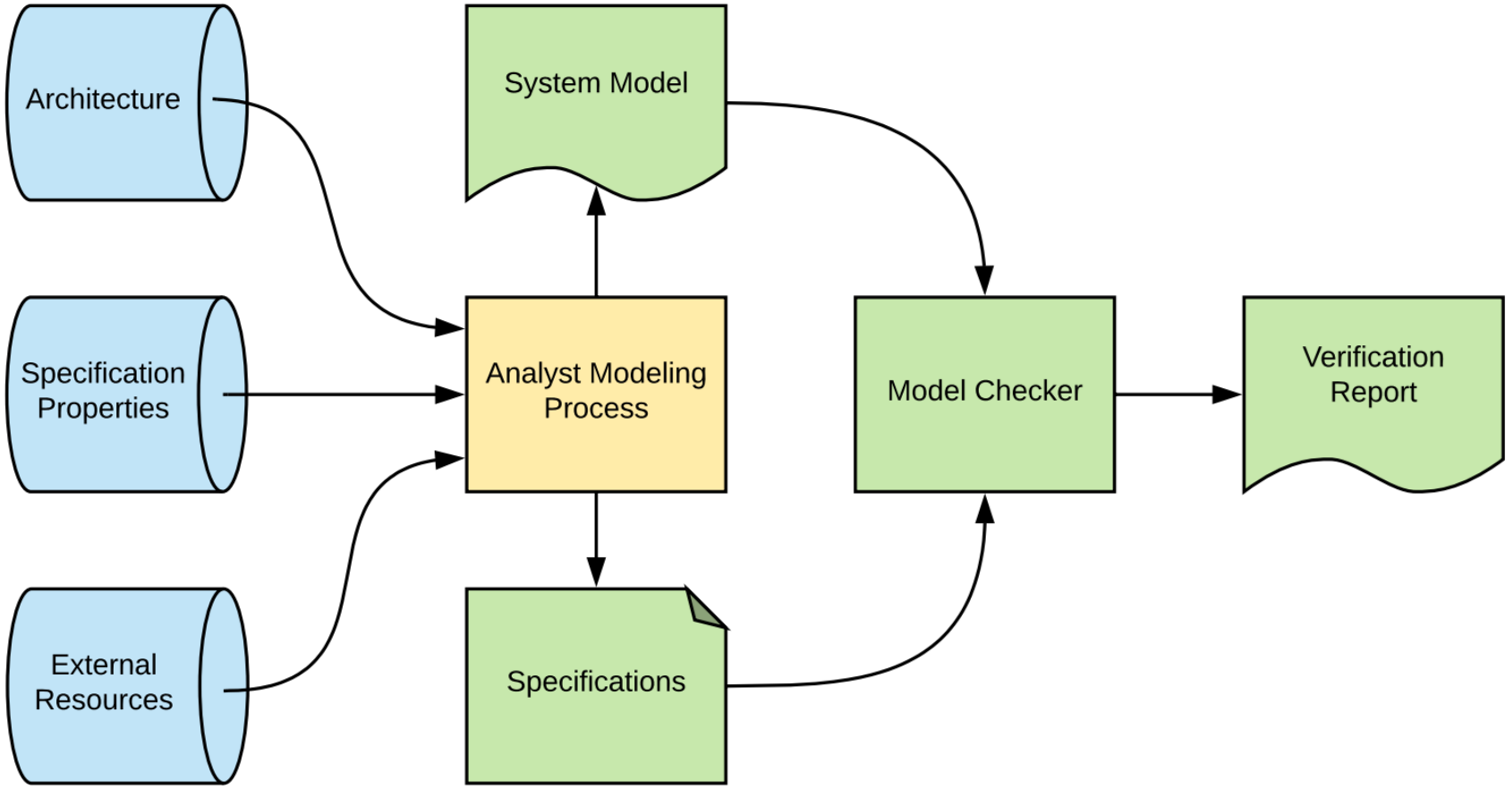
Formulate a framework that:

- extends Degani and Heymann (2002)'s architecture to a new domain
- can discover unanticipated cybersecurity "vulnerabilities"
- captures developments in a theoretically-grounded, application-agnostic approach

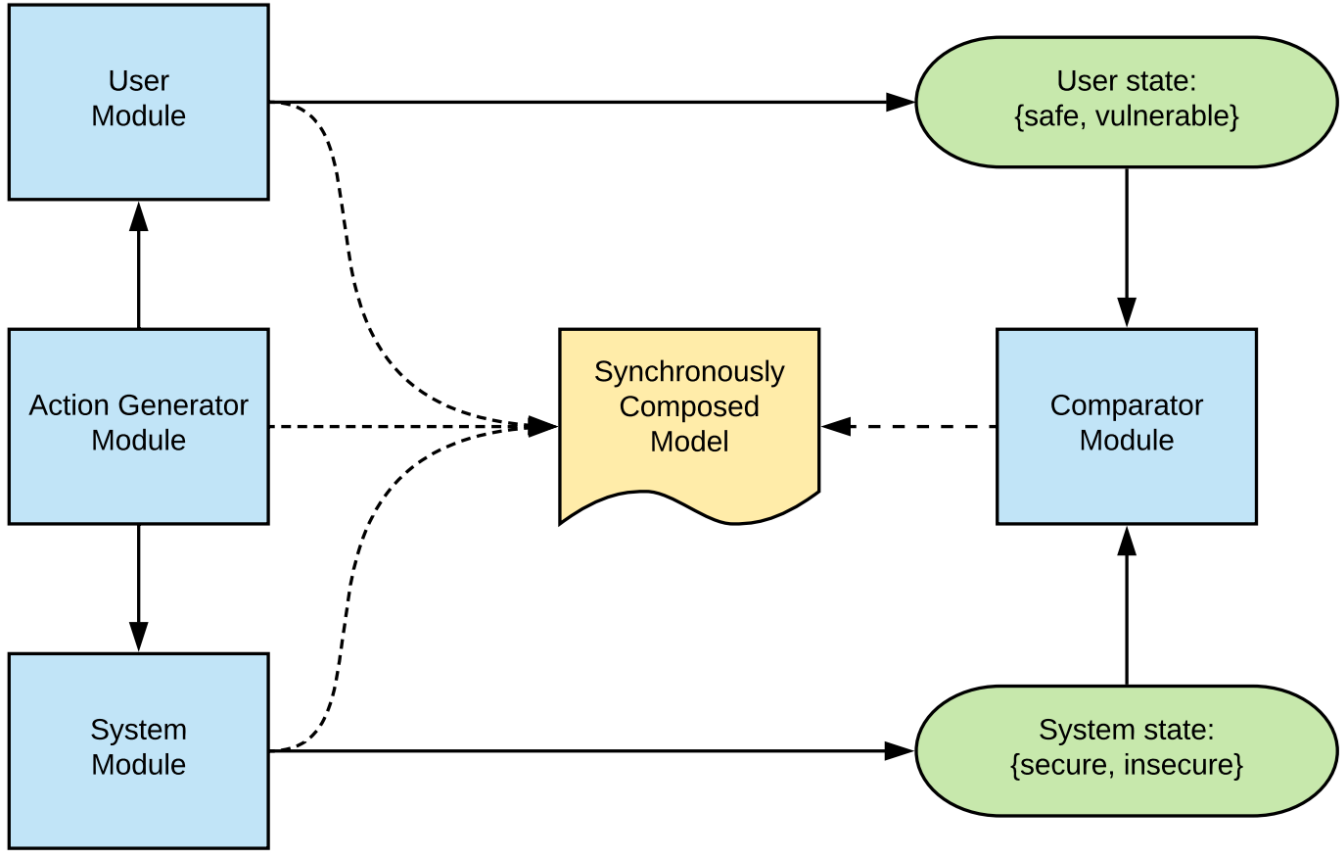
Solution: our Generic Method



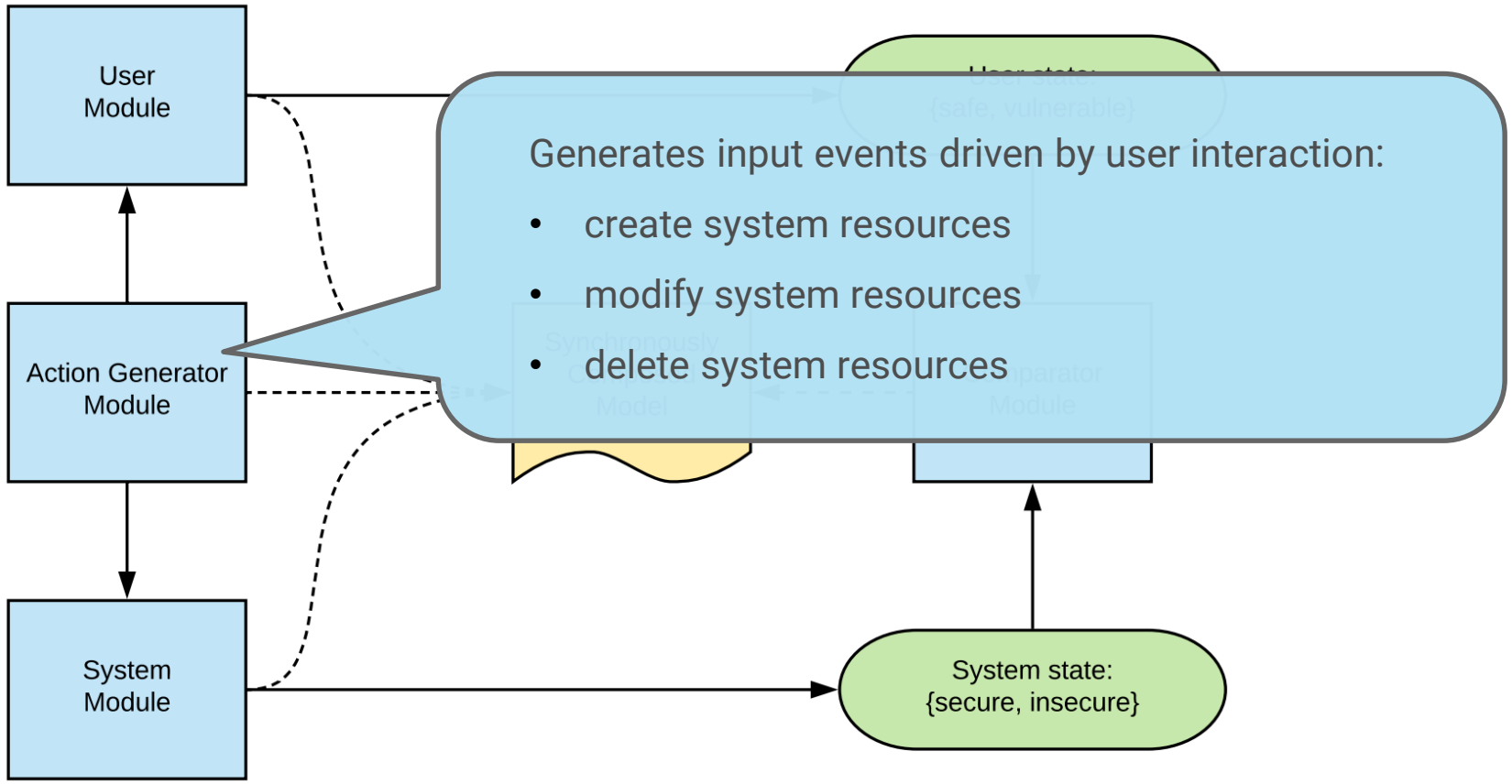
Generic formulation of the method



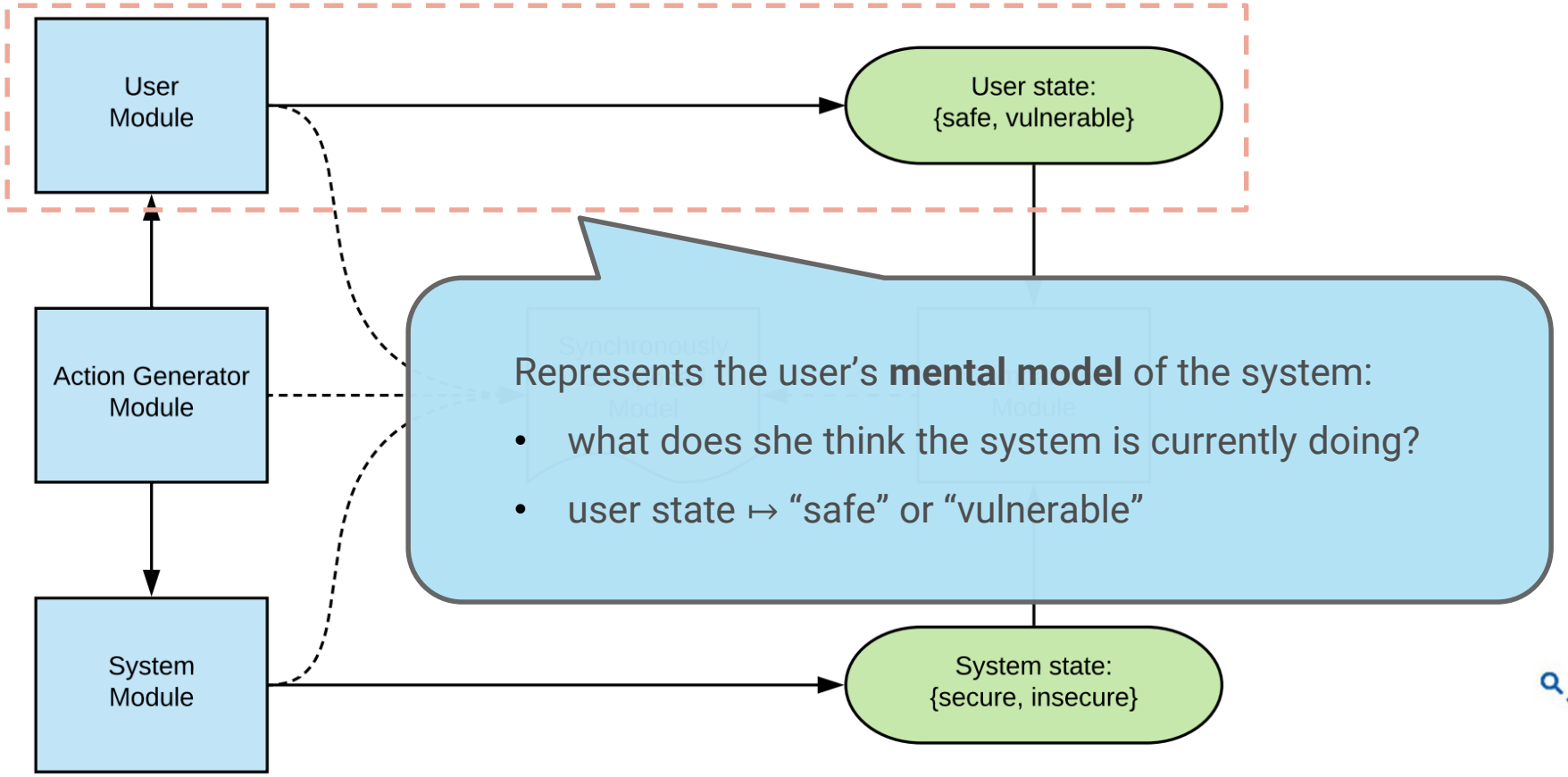
Generic system model architecture



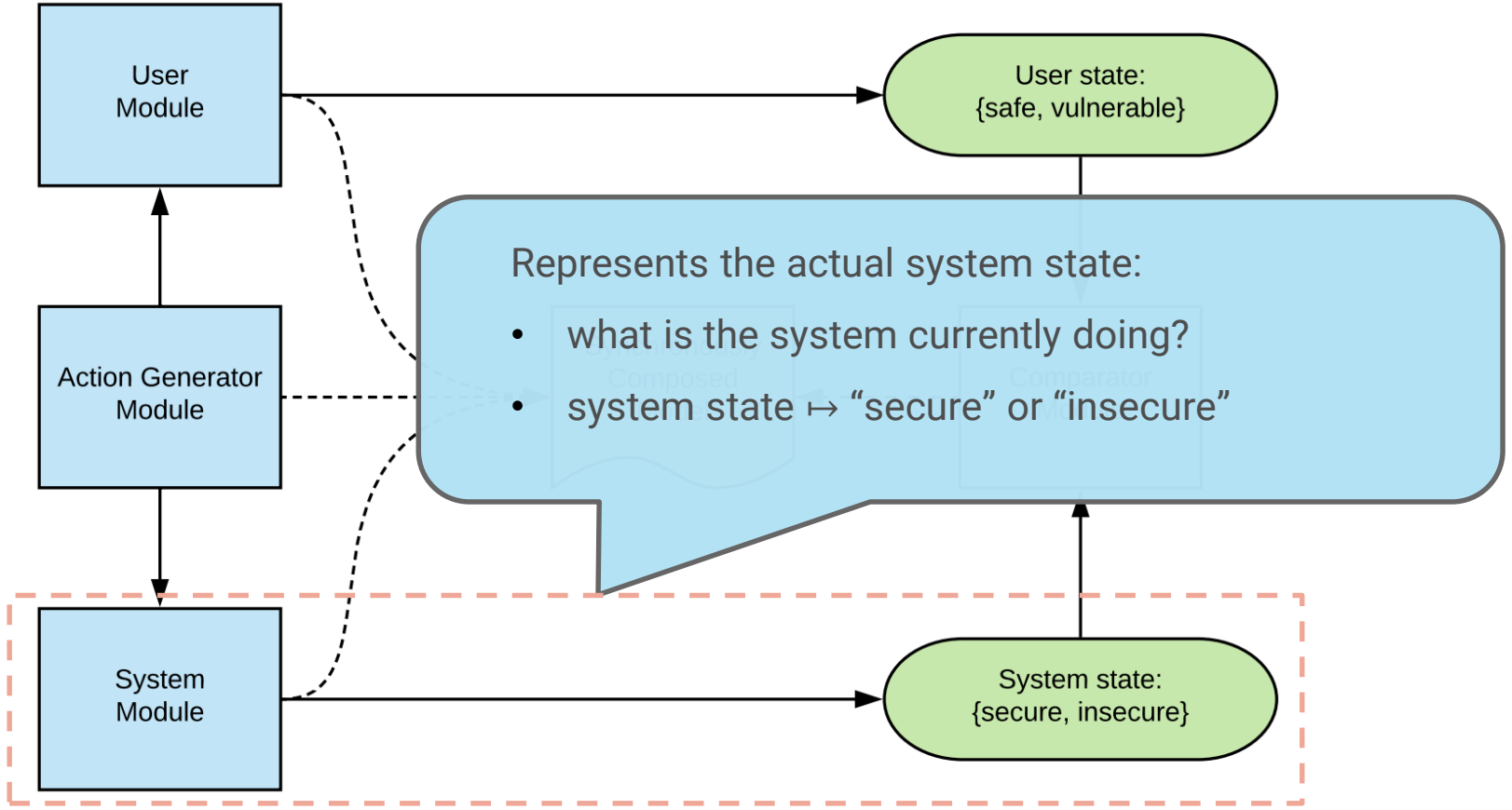
Generic system model architecture



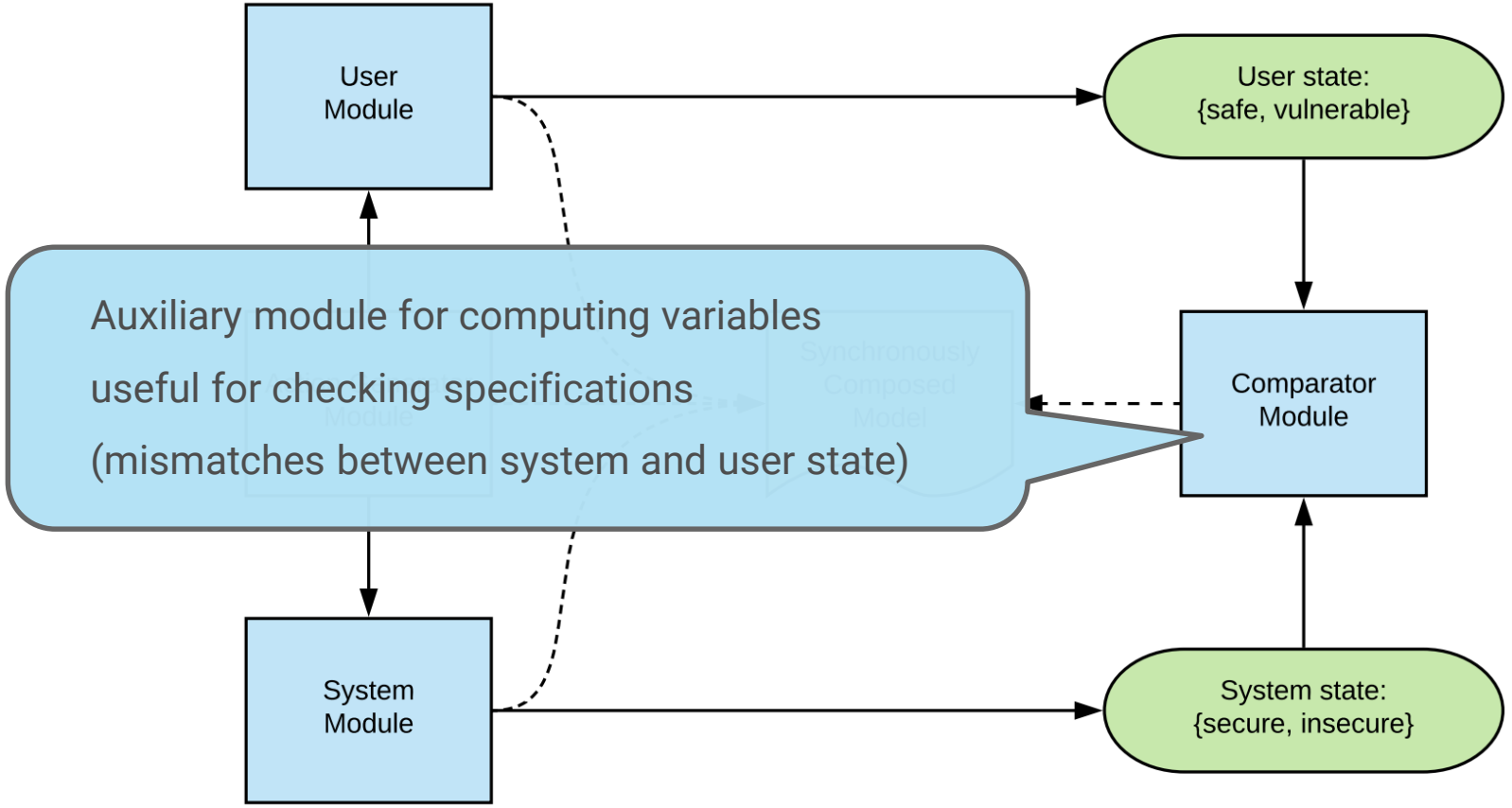
Generic system model architecture



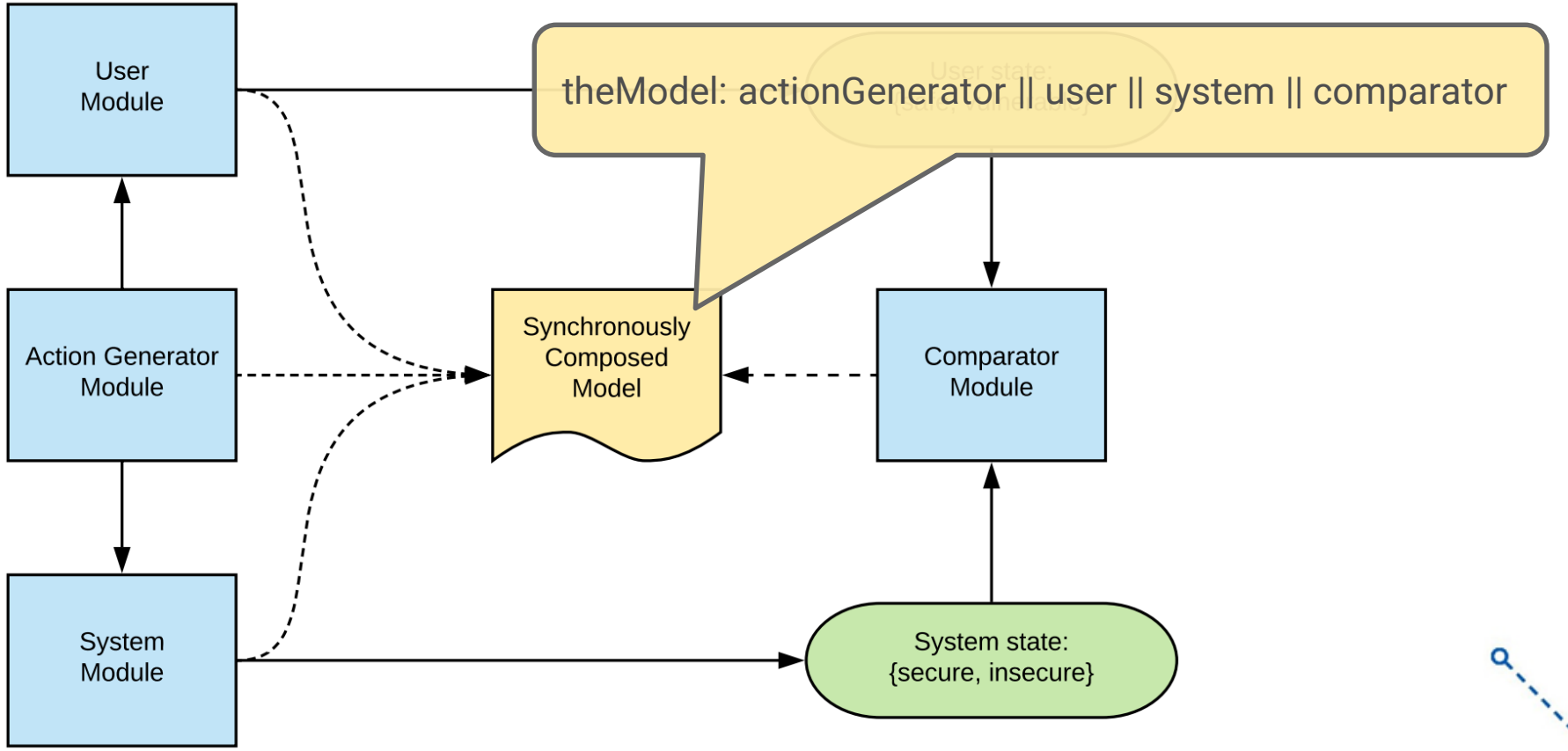
Generic system model architecture



Generic system model



Generic system model



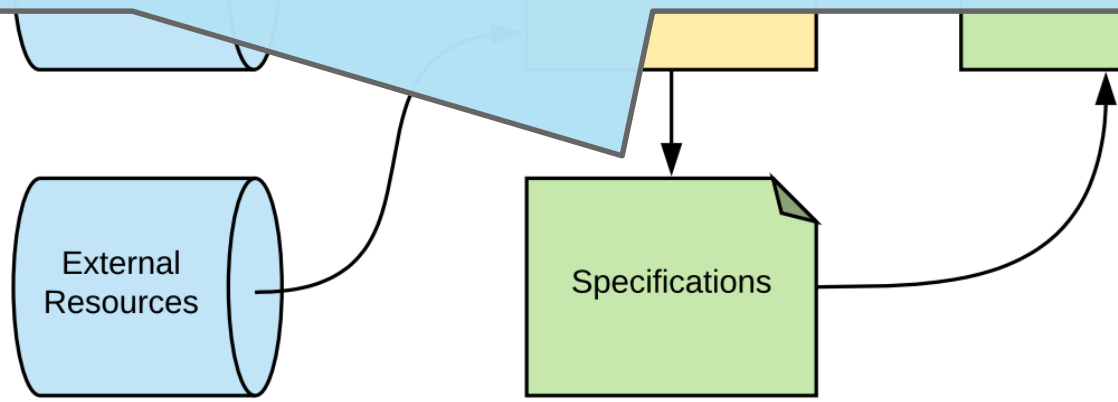
Generic formulation of the method

No Error States:

No False Vulnerability $G \neg (user = vulnerable \wedge system = secure)$

No False Safety $G \neg (user = safe \wedge system = insecure)$

No Blocking States: $G \neg \begin{pmatrix} (user = vulnerable \wedge system = insecure) \\ \text{AND } \mathbf{X}\,(user = vulnerable \wedge system = secure) \end{pmatrix}$



University at Buffalo The State University of New York

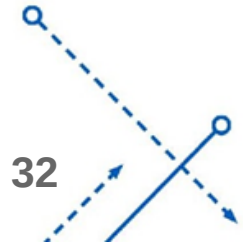
Use case: Amazon Web Services

What is AWS?

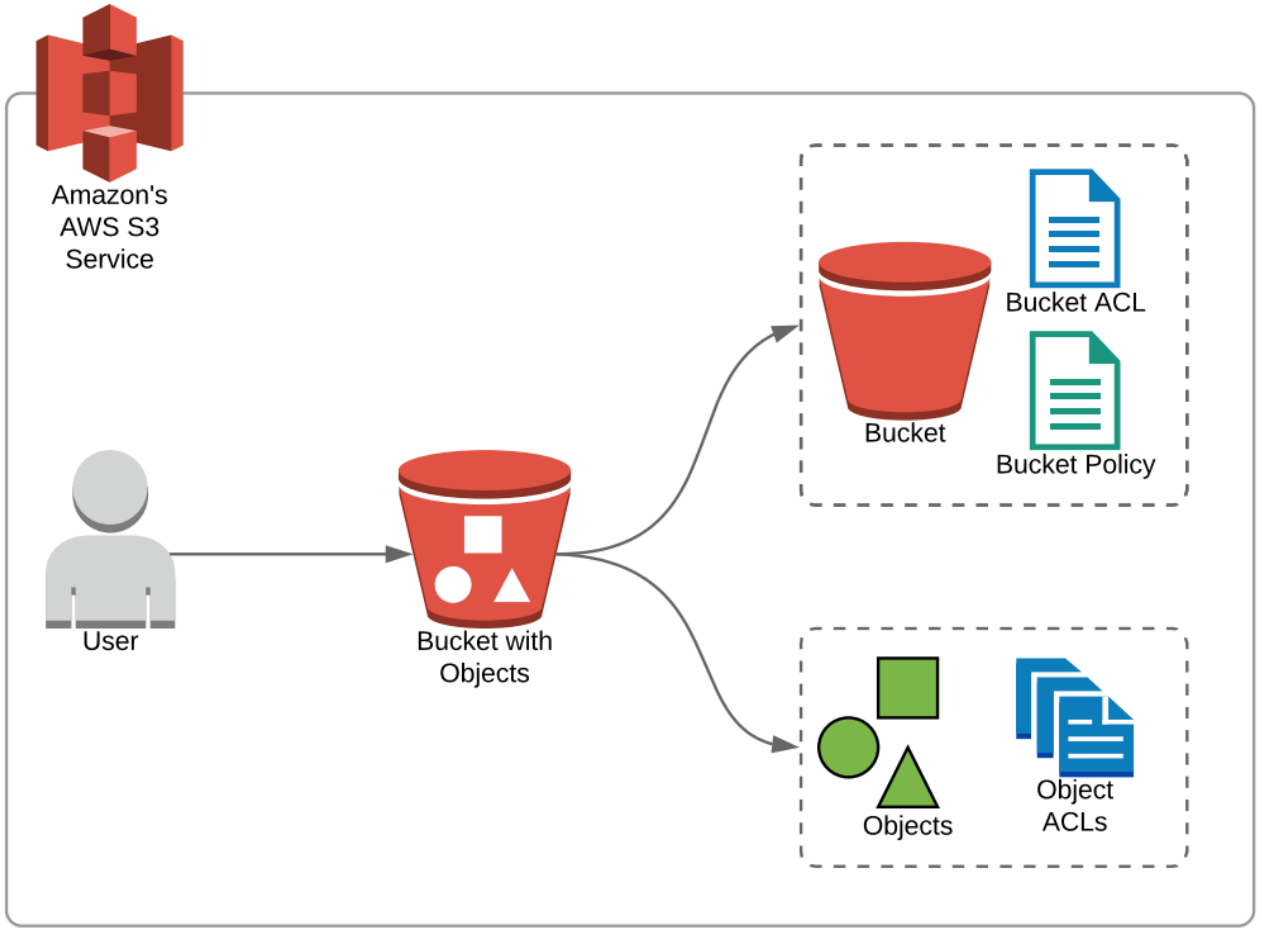
- Largest single provider of cloud computing and data storage services on the planet
- Range of (131) services, from small (AWS EC2) to very large (AWS Snowmobile)
- Netflix, Hulu, GE Oil and Gas, Kellogg's, Airbnb, NASA, NASDAQ, FICO
  - $20B in revenue for Amazon in AY 2017
- Pay as you go, so anyone can create a free account
- What could go wrong?

Customer data “fire sales”

- AWS Simple Storage Service, or S3
- Potentially difficult to secure, leaving data unprotected – easy targets
- Notable recent events:
  - Analytics company, 198 million voting records (Newman, 2017)
  - BAH, 60K DoD files and passwords (Cameron, 2017)
  - Pentagon, 100GB of contract and sensitive info (O’Sullivan, 2017)
  - Verizon, 14 million customer records (Whittaker, 2017)
  - National Credit Union Federation, 100GB of Social Security, bank accounts, and customer credit reports (Ashok, 2017)



How does S3 work?



How does an ACL work?

Access for your AWS account

Account	List objects	Write objects	Read bucket permissions	Write bucket permissions
[redacted]	Yes	Yes	Yes	Yes

Access for other AWS accounts

+ Add account Delete

Account	List objects	Write objects	Read bucket permissions	Write bucket permissions

Public access

Group	List objects	Write objects	Read bucket permissions	Write bucket permissions
Everyone	-	-	-	-

S3 log delivery group

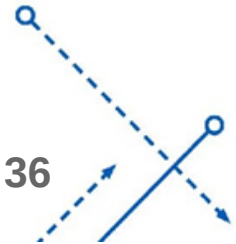
Group	List objects	Write objects	Read bucket permissions	Write bucket permissions
Log Delivery	-	-	-	-

How does a policy work?

```
{
  "Id": "Policy1523476503635",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1523476501084",
      "Action": [
        "s3:GetBucketPolicy",
        "s3:GetObject",
        "s3:PutBucketPolicy",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::alice-taxreturns/*",
      "Principal": "*"
    }
  ]
}
```

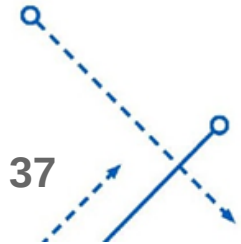
What can you do with S3?

- Store files of any type, from personal to corporate (I do)
- Control who can access what with ACLs and policies
- Apply server-side encryption to keep data safe
- Treat like an unlimited Dropbox or file sharing service
- Host static websites (I did)



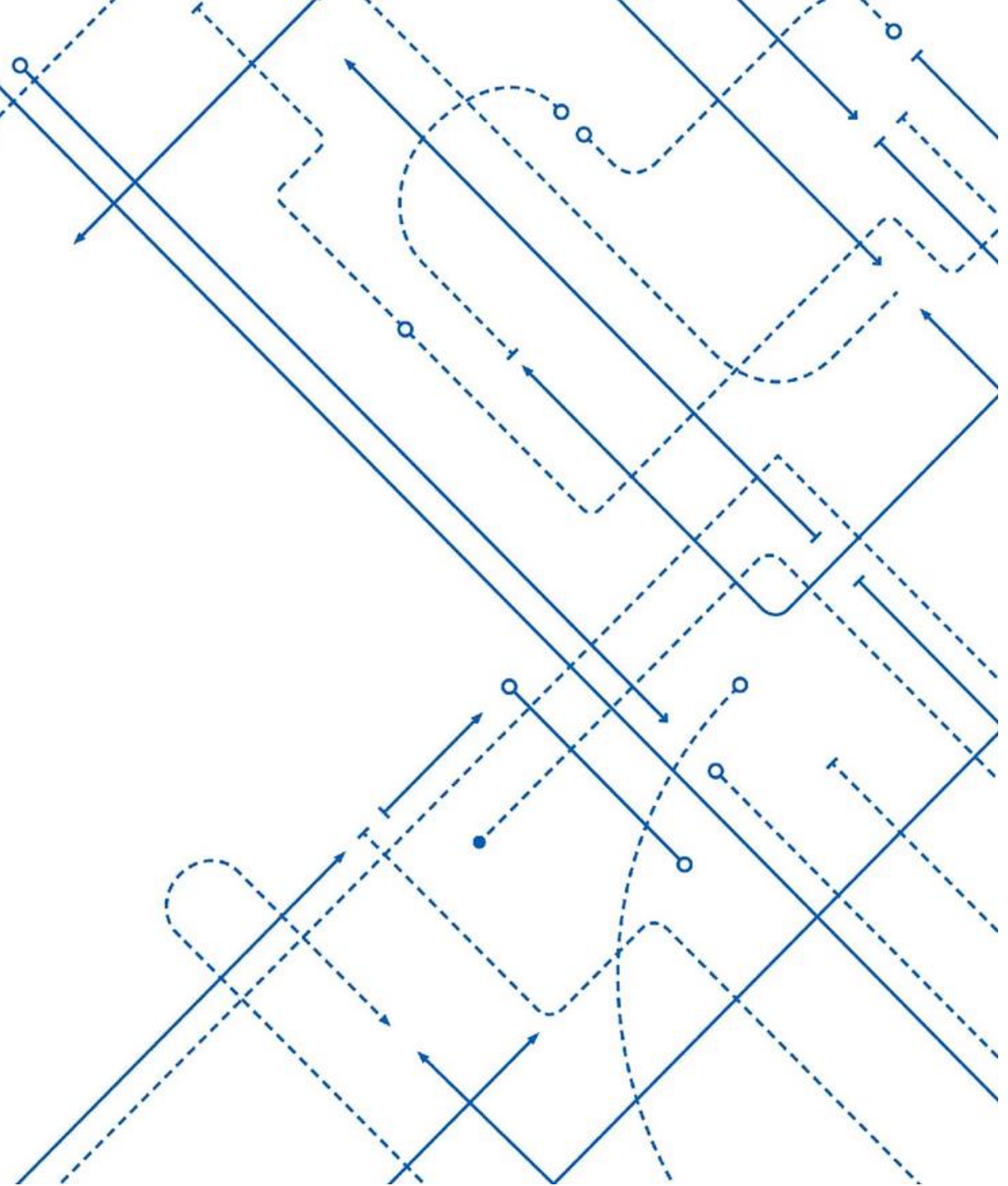
Our method is well-suited for this use case

- Widespread use among a range of customers
- Security controls are difficult to understand, hard to configure, offer overlapping functionality
- Potentially simple user model, potentially complex system model
  - How will users know if something is in an unsafe state?
  - Will they know how to recover if something bad happens?
- Difficult to anticipate how configuration could compromise security



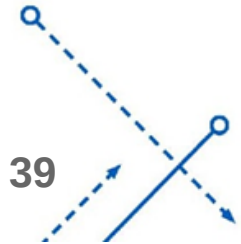
University at Buffalo The State University of New York

Analysis and Results



Analysis scenario

- A user can take a number of actions with their S3 account
  - Create, modify, or delete resources (bucket and objects)
  - Change encryption settings on resources
  - Change permissions on resources (read, write, readPerms, writePerms)
- Can a user interacting with AWS S3 inadvertently compromise their security through a combination of configuration decisions?



```
mental: MODULE =  BEGIN
    LOCAL     thinkBucketExists:       BOOLEAN
    LOCAL     thinkBucketEncryption: encryptionSetting
    LOCAL     thinkBucketACL:          permissionSetting
    LOCAL     thinkObjectExists:       BOOLEAN
    LOCAL     thinkObjectEncryption: encryptionSetting
    LOCAL     thinkObjectACL:          permissionSetting
    INPUT     action:                  userAction
    INPUT     actionPerm:              permissionSetting
    INPUT     actionEnc:               encryptionSetting
    OUTPUT    userWrite:               userState
    OUTPUT    userRead:                userState

    INITIALIZATION
        thinkBucketExists       = FALSE;
        thinkObjectExists       = FALSE;
        thinkBucketEncryption   = none_Option;
        thinkObjectEncryption   = none_Option;

    DEFINITION
        userWrite = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.write OR thinkObjectACL.writePermissions) AND
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
        userRead  = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.read OR thinkObjectACL.readPermissions) AND
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
    …
    …
    TRANSITION  [
        action = create_Object AND thinkBucketExists AND NOT thinkObjectExists -->
            thinkObjectExists'        =  TRUE;
            thinkObjectEncryption'    =  actionEnc;
            thinkObjectACL'           =  actionPerm;

        [] action = delete_Object AND thinkObjectExists -->
            thinkObjectExists'        =  FALSE;
            thinkObjectEncryption'    =  none_Option;

        [] action = change_ObjectPermissions AND thinkObjectExists -->
            thinkObjectACL'           =  actionPerm;

        [] action = change_ObjectEncryption AND thinkObjectExists -->
            thinkObjectEncryption'    =  actionEnc;

        [] action = create_Bucket AND NOT thinkBucketExists -->
            thinkBucketExists'        =  TRUE;
            thinkBucketEncryption'    =  actionEnc;
            thinkBucketACL'           =  actionPerm;

        [] action = delete_Bucket AND thinkBucketExists -->
            thinkBucketExists'        =  FALSE;
            thinkBucketEncryption'    =  none_Option;
            thinkObjectExists'        =  FALSE;
            thinkObjectEncryption'    =  none_Option;

        [] action = change_BucketPermissions AND thinkBucketExists -->
            thinkBucketACL'           =  actionPerm;
            thinkObjectACL'           =  actionPerm;

        [] action = change_BucketEncryption AND thinkBucketExists -->
            thinkBucketEncryption'    =  actionEnc;
            thinkObjectEncryption'    =  IF thinkObjectExists = FALSE THEN
                                             none_Option ELSE actionEnc ENDIF;
        [] ELSE -->
            thinkObjectExists' = thinkObjectExists;
    ]
END;
```

```
mental: MODULE = BEGIN
  LOCAL   thinkBucketExists:      BOOLEAN
  LOCAL   thinkBucketEncryption:  encryptionSetting
  LOCAL   thinkBucketACL:         permissionSetting
  LOCAL   thinkObjectExists:      BOOLEAN
  LOCAL   thinkObjectEncryption:  encryptionSetting
  LOCAL   thinkObjectACL:         permissionSetting
  INPUT   action:                 userAction
  INPUT   actionPerm:             permissionSetting
  INPUT   actionEnc:              encryptionSetting
  OUTPUT  userWrite:              userState
  OUTPUT  userRead:               userState

  INITIALIZATION
      thinkBucketExists       = FALSE;
      thinkObjectExists       = FALSE;
      thinkBucketEncryption   = none_Option;
      thinkObjectEncryption   = none_Option;

  DEFINITION
      userWrite = IF thinkObjectExists  AND thinkObjectACL.public AND
          (thinkObjectACL.write OR thinkObjectACL.writePermissions) AND
          thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
      userRead  = IF thinkObjectExists  AND thinkObjectACL.public AND
          (thinkObjectACL.read OR thinkObjectACL.readPermissions) AND
          thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
  ...
```

Variables describing things the human thinks about the system:

- Whether or not the bucket exists
- How the bucket is encrypted (if at all)
- The bucket's ACL permissions
- Whether or not the object exists
- How the object is encrypted (if at all)
- The object's ACL permissions

```
  TRANSITION [
      action = create_Object AND thinkBucketExists AND NOT thinkObjectExists -->
          thinkObjectExists'      = TRUE;
          thinkObjectEncryption'  = actionEnc;
          thinkObjectACL          = actionPerm;
      [] action = delete_Object AND thinkObjectExists -->
          thinkObjectExists'      = FALSE;
          thinkObjectEncryption'  = none_Option;

      [] action = change_ObjectPermissions AND thinkObjectExists -->
          thinkObjectACL          = actionPerm;

      [] action = change_ObjectEncryption AND thinkObjectExists -->
          thinkObjectEncryption'  = actionEnc;

      [] action = create_Bucket AND NOT thinkBucketExists -->
          thinkBucketExists'      = TRUE;
          thinkBucketEncryption'  = actionEnc;
          thinkBucketACL          = actionPerm;
      [] action = delete_Bucket AND thinkBucketExists -->
          thinkBucketExists'      = FALSE;
          thinkObjectExists'      = FALSE;
          thinkObjectEncryption'  = none_Option;

      [] action = change_BucketPermissions AND thinkBucketExists -->
          thinkBucketACL          = actionPerm;
          thinkObjectACL          = actionPerm;

      [] action = change_BucketEncryption AND thinkBucketExists -->
          thinkBucketEncryption'  = actionEnc;
          thinkObjectEncryption'  = IF thinkObjectExists = FALSE THEN
                                        none_Option ELSE actionEnc ENDIF;
      [] ELSE -->
          thinkObjectExists' = thinkObjectExists;
      ]
  END;
```

```
mental: MODULE =  BEGIN
    LOCAL     thinkBucketExists:       BOOLEAN
    LOCAL     thinkBucketEncryption: encryptionSetting
    LOCAL     thinkBucketACL:          permissionSetting
    LOCAL     thinkObjectExists:       BOOLEAN
    LOCAL     thinkObjectEncryption: encryptionSetting
    LOCAL     thinkObjectACL:          permissionSetting
    INPUT     action:                  userAction
    INPUT     actionPerm:              permissionSetting
    INPUT     actionEnc:               encryptionSetting
    OUTPUT    userWrite:               userState
    OUTPUT    userRead:                userState

    INITIALIZATION
        thinkBucketExists       = FALSE;
        thinkObjectExists       = FALSE;
        thinkBucketEncryption   = none_Option;
        thinkObjectEncryption   = none_Option;

    DEFINITION
        userWrite = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.write OR thinkObjectACL.writePermissions) AND
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
        userRead  = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.read OR thinkObjectACL.readPermissions) AND
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
    …
```

Variables describing actions and their associated changes to ACL permissions and encryption

```
    …
    TRANSITION  [
        action = create_Object AND thinkBucketExists AND NOT thinkObjectExists -->
            thinkObjectExists'        = TRUE;
            thinkObjectEncryption'    = actionEnc;
            thinkObjectACL'           = actionPerm;

        [] action = delete_Object AND thinkObjectExists -->
            thinkObjectExists'        = FALSE;
            thinkObjectEncryption'    = none_Option;

        [] action = change_ObjectPermissions AND thinkObjectExists -->
            thinkObjectACL'           = actionPerm;

        [] action = change_ObjectEncryption AND thinkObjectExists -->
            thinkObjectEncryption'    = actionEnc;

        [] action = create_Bucket AND NOT thinkBucketExists -->
            thinkBucketExists'        = TRUE;
            thinkBucketACL'           = actionPerm;

        [] action = delete_Bucket AND thinkBucketExists -->
            thinkBucketExists'        = FALSE;
            thinkBucketEncryption'    = none_Option;
            thinkObjectExists'        = FALSE;
            thinkObjectEncryption'    = none_Option;

        [] action = change_BucketPermissions AND thinkBucketExists -->
            thinkBucketACL'           = actionPerm;
            thinkObjectACL'           = actionPerm;

        [] action = change_BucketEncryption AND thinkBucketExists -->
            thinkBucketEncryption'    = actionEnc;
            thinkObjectEncryption'    = IF thinkObjectExists = FALSE THEN
                                            none_Option ELSE actionEnc ENDIF;
        [] ELSE -->
            thinkObjectExists' = thinkObjectExists;
    ]
END;
```

```
mental: MODULE =  BEGIN
    LOCAL     thinkBucketExists:       BOOLEAN
    LOCAL     thinkBucketEncryption: encryptionSetting
    LOCAL     thinkBucketACL:          permissionSetting
    LOCAL     thinkObjectExists:       BOOLEAN
    LOCAL     thinkObjectEncryption: encryptionSetting
    LOCAL     thinkObjectACL:          permissionSetting
    INPUT     action:                  userAction
    INPUT     actionPerm:              permissionSetting
    INPUT     actionEnc:               encryptionSetting
    OUTPUT    userWrite:               userState
    OUTPUT    userRead:                userState

    INITIALIZATION
        thinkBucketExists       = FALSE;
        thinkObjectExists       = FALSE;
        thinkBucketEncryption   = none_Option;
        thinkObjectEncryption   = none_Option;

    DEFINITION
        userWrite = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.write OR thinkObjectACL.writePermissions) AND
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
        userRead  = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.read OR thinkObjectACL.readPermissions) AND
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
    ...
```

Variables representing what the human thinks about the Read and Write state of the system: safe or vulnerable

```
    ...
    TRANSITION  [
        action = create_Object AND thinkBucketExists AND NOT thinkObjectExists -->
            thinkObjectExists'        = TRUE;
            thinkObjectEncryption'  = actionEnc;
            thinkObjectACL'           = actionPerm;

        [] action = delete_Bucket AND thinkBucketExists -->
            thinkBucketExists'        = FALSE;
            thinkBucketEncryption'  = none_Option;
            thinkObjectExists'        = FALSE;
            thinkObjectEncryption'  = none_Option;

        [] action = change_BucketPermissions AND thinkBucketExists -->
            thinkBucketACL'           = actionPerm;
            thinkObjectACL'           = actionPerm;

        [] action = change_BucketEncryption AND thinkBucketExists -->
            thinkBucketEncryption'  = actionEnc;
            thinkObjectEncryption'  = IF thinkObjectExists = FALSE THEN
                                          none_Option ELSE actionEnc ENDIF;
        [] ELSE -->
            thinkObjectExists' = thinkObjectExists;
    ]
END;
```

```
mental: MODULE =  BEGIN
    LOCAL     thinkBucketExists:       BOOLEAN
    LOCAL     thinkBucketEncryption: encryptionSetting
    LOCAL     thinkBucketACL:          permissionSetting
    LOCAL     thinkObjectExists:       BOOLEAN
    LOCAL     thinkObjectEncryption: encryptionSetting
    LOCAL     thinkObjectACL:          permissionSetting
    INPUT     action:                  userAction
    INPUT     actionPerm:              permissionSetting
    INPUT     actionEnc:               encryptionSetting
    OUTPUT    userWrite:               userState
    OUTPUT    userRead:                userState

    INITIALIZATION
        thinkBucketExists       = FALSE;
        thinkObjectExists       = FALSE;
        thinkBucketEncryption   = none_Option;
        thinkObjectEncryption   = none_Option;

    DEFINITION
        userWrite = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.write OR thinkObjectACL.writePermissions) A
            thinkObjectEncryption = none_Option THEN vulnerable ELSE s
        userRead  = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.read OR thinkObjectACL.readPermissions) AN
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe
    ...
```

```
...
TRANSITION  [
    action = create_Object AND thinkBucketExists AND NOT thinkObjectExists -->
        thinkObjectExists'        = TRUE;
        thinkObjectEncryption'    = actionEnc;
        thinkObjectACL'           = actionPerm;

    [] action = delete_Object AND thinkObjectExists -->
        thinkObjectExists'        = FALSE;
        thinkObjectEncryption'    = none_Option;

    [] action = change_ObjectPermissions AND thinkObjectExists -->
        thinkObjectACL'           = actionPerm;

    [] action = change_ObjectEncryption AND thinkObjectExists -->
        thinkObjectEncryption'    = actionEnc;

    [] action = create_Bucket AND NOT thinkBucketExists -->
        thinkBucketExists'        = TRUE;
        thinkBucketEncryption'    = actionEnc;
        thinkBucketACL'           = actionPerm;
    ...
    END;
```

The initial mental model state:

- No bucket or object exists
- No encryption is set
- No assumptions are made about permissions

Guarded transitions in mental model state based on actions

```
m
LOCAL     thinkBucketEncryption: encryptionSetting
LOCAL     thinkObjectExists:     BOOLEAN
LOCAL     thinkObjectEncryption: encryptionSetting
LOCAL     thinkObjectACL:        permissionSetting
INPUT     action:                userAction
INPUT     actionPerm:            permissionSetting
INPUT     actionEnc:             encryptionSetting
OUTPUT    userWrite:             userState
OUTPUT    userRead:              userState

INITIALIZATION
    thinkBucketExists       = FALSE;
    thinkObjectExists       = FALSE;
    thinkBucketEncryption   = none_Option;
    thinkObjectEncryption   = none_Option;

DEFINITION
    userWrite = IF thinkObjectExists  AND thinkObjectACL.public AND
        (thinkObjectACL.write OR thinkObjectACL.writePermissions) AND
        thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
    userRead  = IF thinkObjectExists  AND thinkObjectACL.public AND
        (thinkObjectACL.read OR thinkObjectACL.readPermissions) AND
        thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
...
```

```
    ...
    TRANSITION  [
        action = create_Object AND thinkBucketExists AND NOT thinkObjectExists -->
            thinkObjectExists'        = TRUE;
            thinkObjectEncryption'    = actionEnc;
            thinkObjectACL'           = actionPerm;

        [] action = delete_Object AND thinkObjectExists -->
            thinkObjectExists'        = FALSE;
            thinkObjectEncryption'    = none_Option;

        [] action = change_ObjectPermissions AND thinkObjectExists -->
            thinkObjectACL'           = actionPerm;

        [] action = change_ObjectEncryption AND thinkObjectExists -->
            thinkObjectEncryption'    = actionEnc;

        [] action = create_Bucket AND NOT thinkBucketExists -->
            thinkBucketExists'        = TRUE;
            thinkBucketEncryption'    = actionEnc;
            thinkBucketACL'           = actionPerm;

        [] action = delete_Bucket AND thinkBucketExists -->
            thinkBucketExists'        = FALSE;
            thinkBucketEncryption'    = none_Option;
            thinkObjectExists'        = FALSE;
            thinkObjectEncryption'    = none_Option;

        [] action = change_BucketPermissions AND thinkBucketExists -->
            thinkBucketACL'           = actionPerm;
            thinkObjectACL'           = actionPerm;

        [] action = change_BucketEncryption AND thinkBucketExists -->
            thinkBucketEncryption'    = actionEnc;
            thinkObjectEncryption'    = IF thinkObjectExists = FALSE THEN
                                            none_Option ELSE actionEnc ENDIF;
        [] ELSE -->
            thinkObjectExists' = thinkObjectExists;
    ]
END;
```

```
mental: MODULE =  BEGIN
    LOCAL    thinkBucketExists:        BOOLEAN
    LOCAL    thinkBucketEncryption: encryptionSetting
    LOCAL    thinkBucketACL:           permissionSetting
    LOCAL    thinkObjectExists:        BOOLEAN
    LOCAL    thinkObjectEncryption: encryptionSetting
    LOCAL    thinkObjectACL:           permissionSetting
    INPUT    action:                   userAction
    INPUT    actionPerm:               permissionSetting
    INPUT    actionEnc:                encryptionSetting
    OUTPUT   userWrite:                userState
    OUTPUT   userRead:                 userState

    INITIALIZATION
        thinkObjectExists        = FALSE;
        thinkBucketEncryption = none_Option;
        thinkObjectEncryption = none_Option;
    DEFINITION
        userWrite = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.write OR thinkObjectACL.writePermissions) AND
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
        userRead  = IF thinkObjectExists  AND thinkObjectACL.public AND
            (thinkObjectACL.read OR thinkObjectACL.readPermissions) AND
            thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
    ...
```

Example: Changing the bucket permissions sets the bucket's and the object's permissions to the new permissions values

```
    ...
    TRANSITION  [
        action = create_Object AND thinkBucketExists AND NOT thinkObjectExists -->
            thinkObjectExists'        = TRUE;
            thinkObjectEncryption'  = actionEnc;
            thinkObjectACL'           = actionPerm;

        [] action = delete_Object AND thinkObjectExists -->
            thinkObjectExists'        = FALSE;
            thinkObjectEncryption'  = none_Option;

        [] action = change_ObjectPermissions AND thinkObjectExists -->
            thinkObjectACL'           = actionPerm;

        [] action = change_ObjectEncryption AND thinkObjectExists -->
            thinkObjectEncryption'  = actionEnc;

        [] action = create_Bucket AND NOT thinkBucketExists -->
            thinkBucketExists'        = TRUE;
            thinkBucketEncryption'  = actionEnc;
            thinkBucketACL'           = actionPerm;

        [] action = delete_Bucket AND thinkBucketExists -->
            thinkBucketExists'        = FALSE;
            thinkBucketEncryption'  = none_Option;
            thinkObjectExists'        = FALSE;
            thinkObjectEncryption'  = none_Option;

        [] action = change_BucketPermissions AND thinkBucketExists -->
            thinkBucketACL'           = actionPerm;
            thinkObjectACL'           = actionPerm;

        [] action = change_BucketEncryption AND thinkBucketExists -->
            thinkBucketEncryption'  = actionEnc;
            thinkObjectEncryption'  = IF thinkObjectExists = FALSE THEN
                                          none_Option ELSE actionEnc ENDIF;
        [] ELSE -->
            thinkObjectExists' = thinkObjectExists;
    ]
END;
```

The human's perception about Read and Write safety:

IF the object exists and: it has public permissions,
the object can be written or its permissions can be written
it has no encryption

THEN then writing is Vulnerable **OTHERWISE** writing is Safe

IF the object exists and: it has public permissions,
the object can be read or its permissions can be read
it has no encryption

THEN then reading is Vulnerable **OTHERWISE** reading is Safe

```
DEFINITION
    userWrite = IF thinkObjectExists  AND thinkObjectACL.public AND
        (thinkObjectACL.write OR thinkObjectACL.writePermissions) AND
        thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
    userRead  = IF thinkObjectExists  AND thinkObjectACL.public AND
        (thinkObjectACL.read OR thinkObjectACL.readPermissions) AND
        thinkObjectEncryption = none_Option THEN vulnerable ELSE safe ENDIF;
                                                                    ]
                                                                END;
...
```

```
[] action = change_BucketEncryption AND thinkBucketExists -->
    thinkBucketEncryption'  =  actionEnc;
    thinkObjectEncryption'  =  IF thinkObjectExists = FALSE THEN
                                   none_Option ELSE actionEnc ENDIF;
[] ELSE -->
    thinkObjectExists' = thinkObjectExists;
```

Results

Table 4.1: Amazon AWS S3 Use Case Verification Results

	False Vulnerability			False Safety			Blocking		
	States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result
Read	932,580	0.12	✘	932,580	0.12	✘	3,229,908	0.61	✔
Write	932,580	0.12	✘	932,580	0.11	✘	3,188,724	0.36	✔

Note. A ✔ indicates a proof was returned, while a ✘ indicates a counterexample was returned.

Results

Table 4.1: Amazon AWS S3 Use Case Verification Results

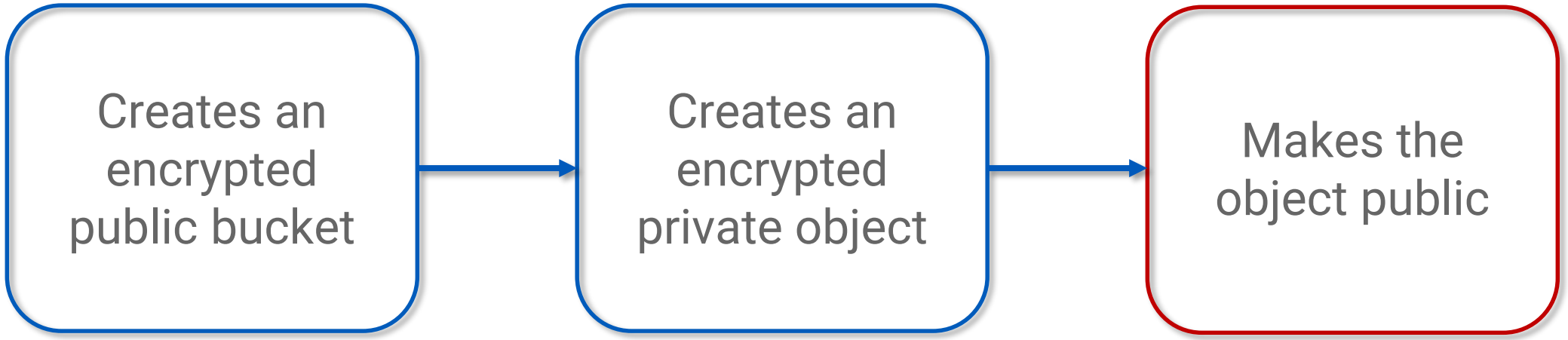
	False Vulnerability			False Safety			Blocking		
	States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result
Read	932,580	0.12	✘	932,580	0.12	✘	3,229,908	0.61	✔
Write	932,580	0.12	✘	932,580	0.11	✘	3,188,724	0.36	✔

Note. A ✔ indicates a proof was returned, while a ✘ indicates a counterexample was returned.

Improper Understanding of Permissions

Improper Understanding of Encryption

Results: False Security Read Mismatch



Results: False Vulnerability Write Mismatch

Validation

- We recreated all four scenarios within an AWS S3 account
- All represent sensible scenarios possible with a minimum of clicks and steps
- Permissions mismatches could explain majority of scenarios identified in press coverage about AWS breaches (more investigation needed)

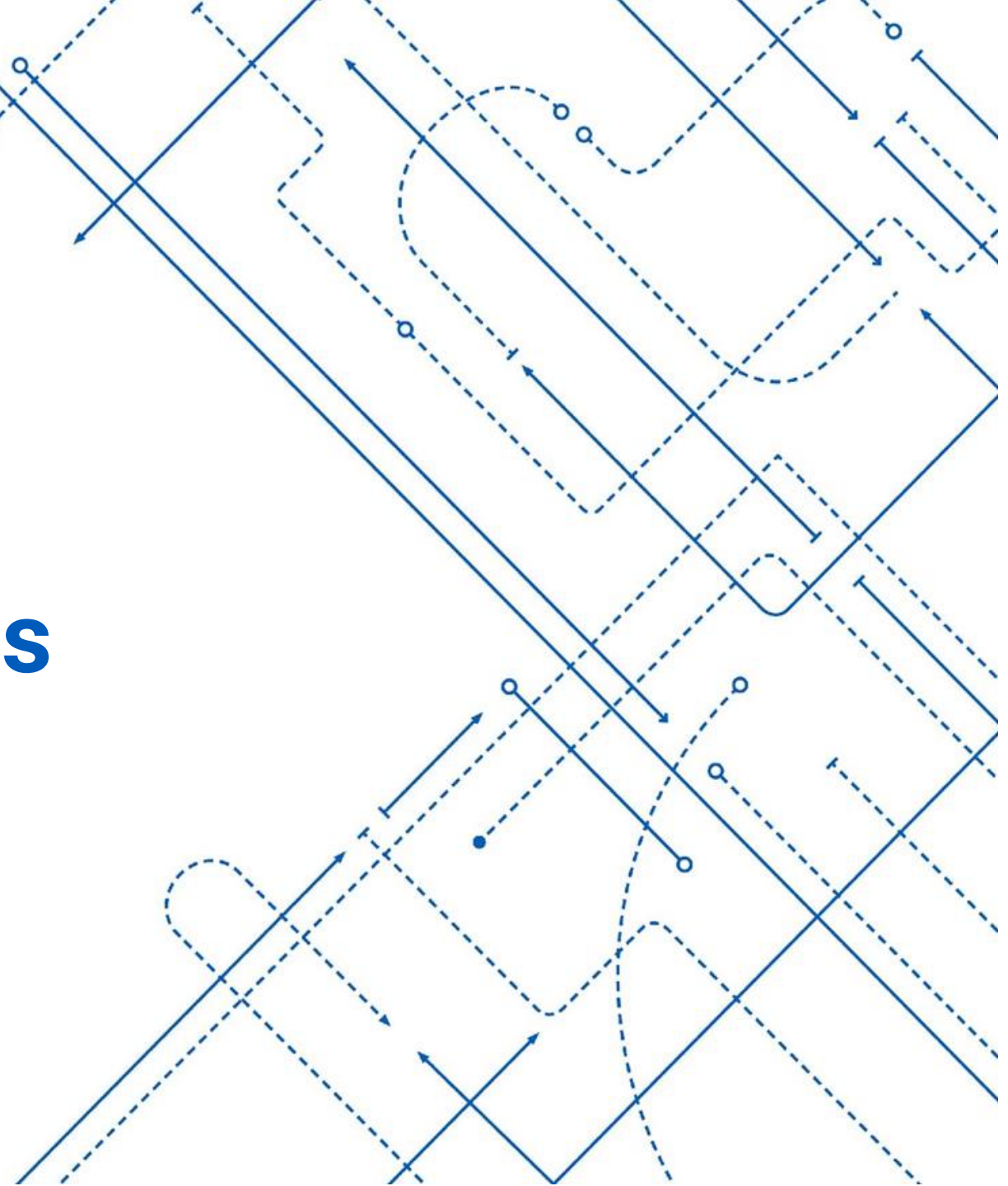
Potential limitations

- Our method only uses a generically-instantiated mental model
- Lacks the capability to determine effects of different mental models on cybersecurity
- Addressed in second part of the dissertation

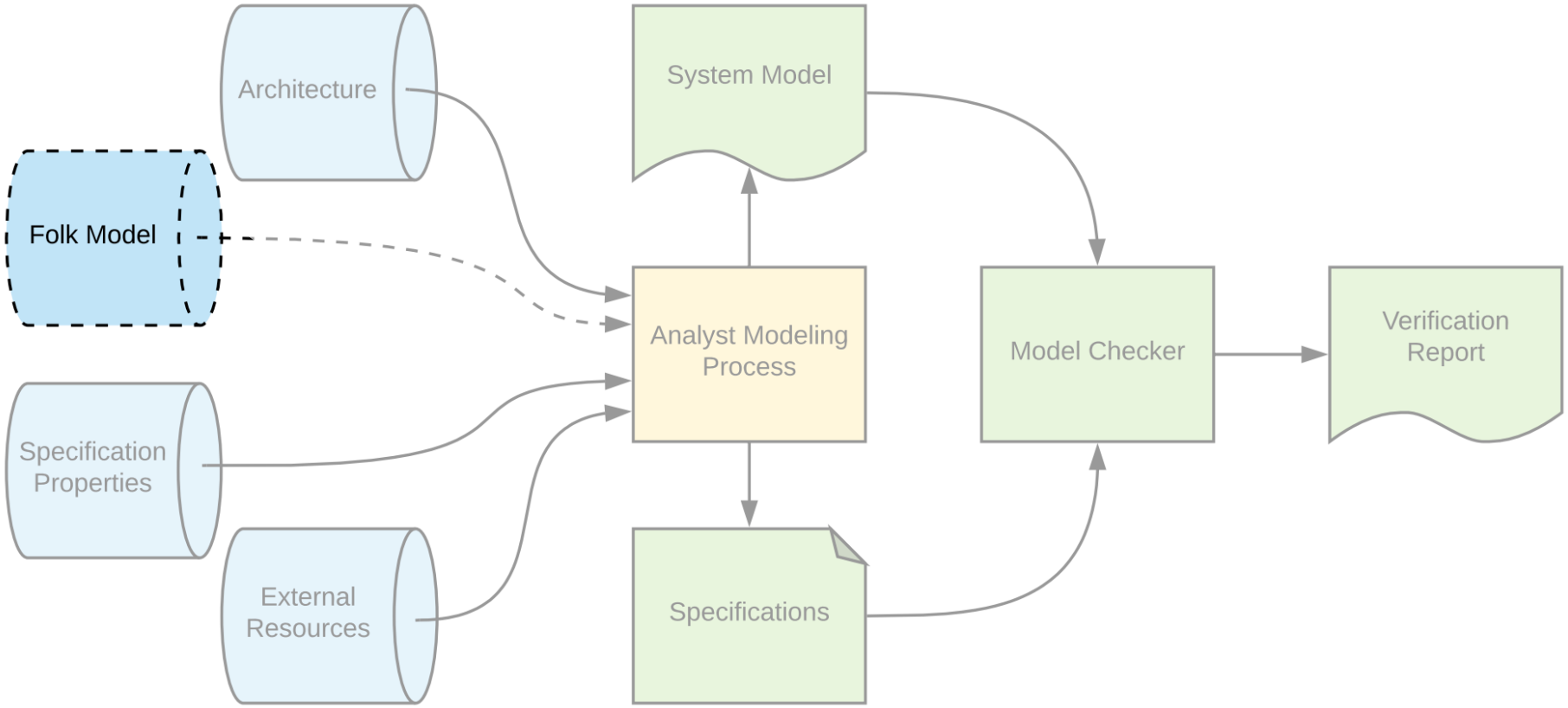
University at Buffalo The State University of New York

Part II:

Integrating folk models of cybersecurity vulnerabilities



Extending the method



Extending the method: Folk model input

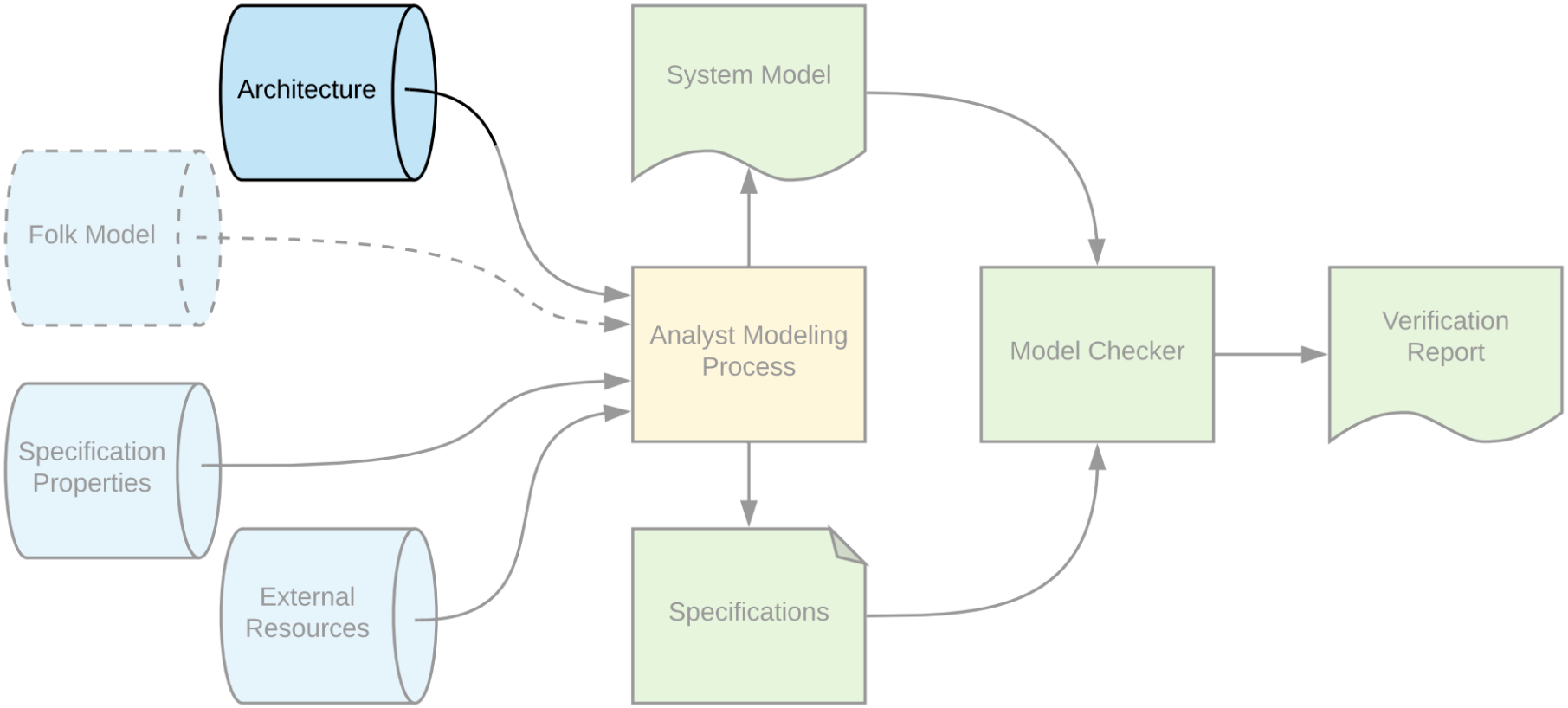
Table 5.1: Folk models and adherence to security advice

Computer security advice	Folk models			
	Graffiti	Burglar	Big Fish	Contractor
1. Use anti-virus software	∅	3	1	1
2. Keep anti-virus software updated	∅	∅	∅	1
3. Regularly scan computer with anti-virus	∅	∅	∅	1
4. Use security software (firewall, etc.)	2	2	2	1
5. Don't click on attachments	3	3	∅	∅
6. Be careful downloading from websites	2	2	1	1
7. Be careful which websites you visit	3	3	2	3
8. Disable scripting in web and mail	∅	∅	∅	1
9. Use good passwords	2	∅	2	1
10. Make regular backups	3	1	1	1
11. Keep patches up-to-date	3	3	1	1
12. Turn off computer when not in use	2	3	1	1

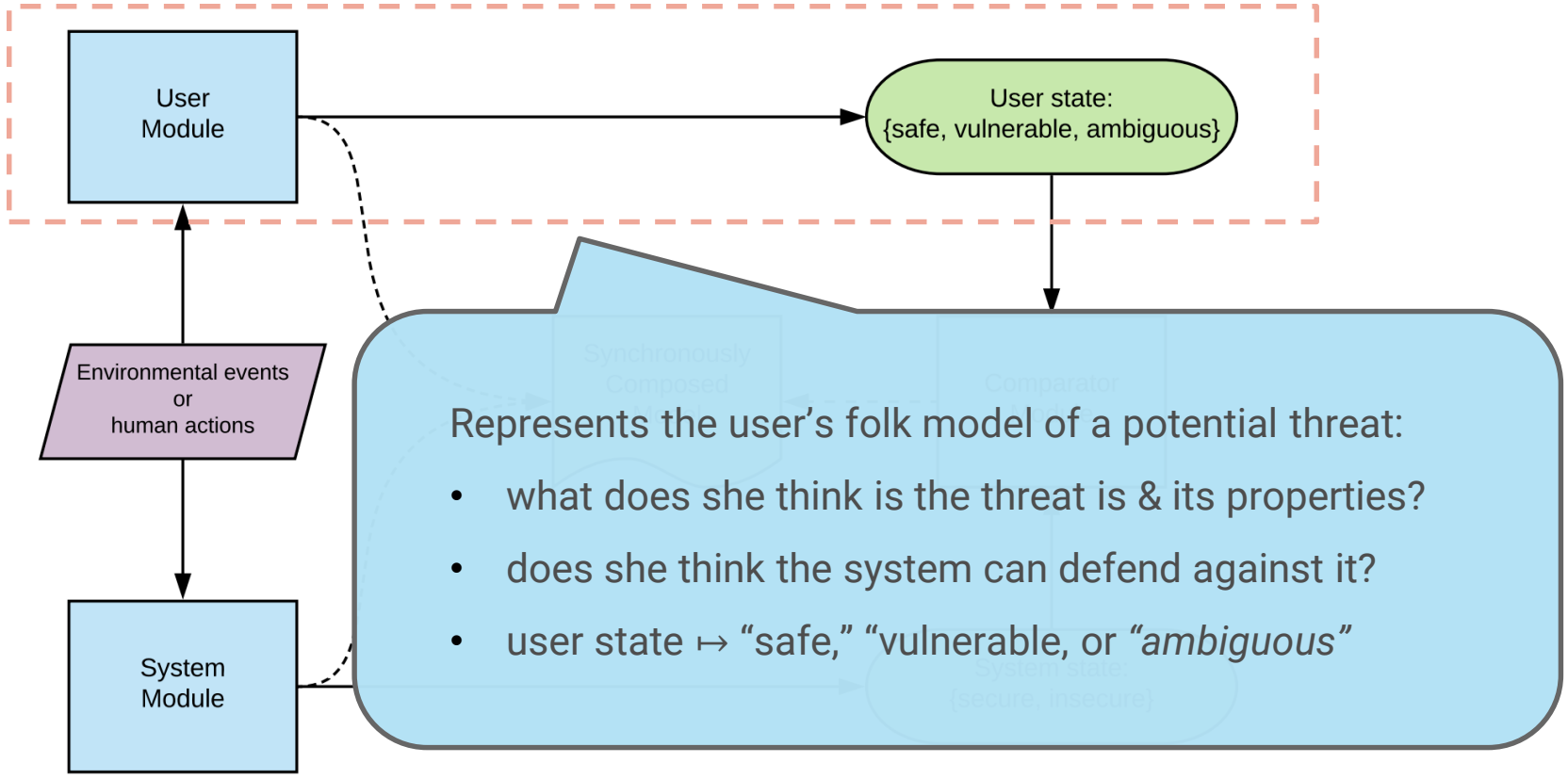
3: "It is very important to follow this advice."
2: "Following this advice might help, but it isn't all that important to do."
1: "It is not necessary to follow this advice."
∅: "This model has nothing to say about this advice, or there is insufficient data from interviews to determine an opinion."

Note. A summary of survey user data regarding the importance of following security advice with respect to their folk models of hackers. Descriptions of the importance levels appearing in quotations has been taken directly from Wash, 2010, p. 10, and the remainder of the table has been adapted.

Extending the method



Extending the method: Updated architecture



Extending the method: Updated architecture

No Error States:

No False Vulnerability

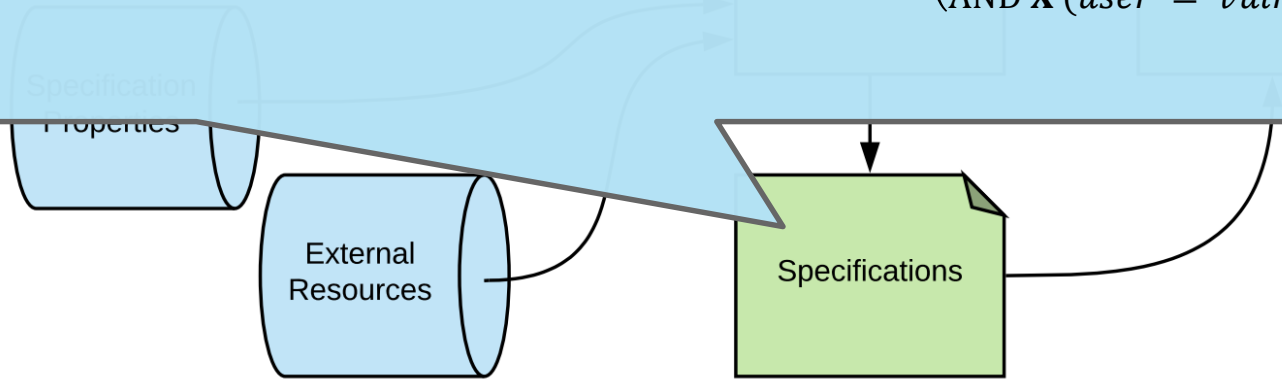
$$G \neg (user = vulnerable \wedge system = secure)$$
$$G \neg (user = ambiguous \wedge system = secure)$$

No False Safety

$$G \neg (user = safe \wedge system = insecure)$$
$$G \neg (user = ambiguous \wedge system = insecure)$$

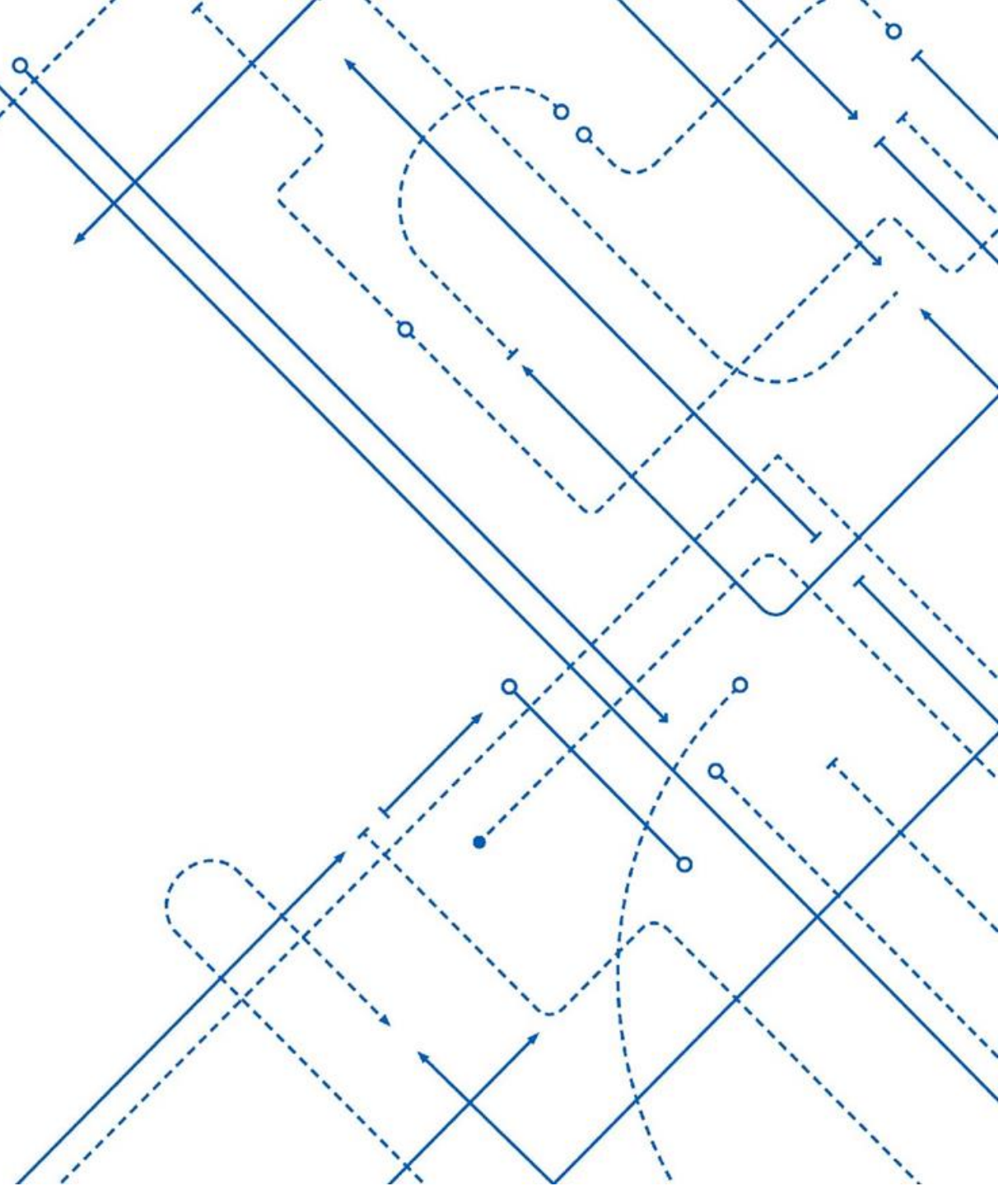
No Blocking States:

$$G \neg \begin{pmatrix} (user = vulnerable \wedge system = insecure) \\ \text{AND } \mathbf{X}\,(user = vulnerable \wedge system = secure) \end{pmatrix}$$



University at Buffalo The State University of New York

Use case: Phishing attacks



Analysis scenario

- Phishing: sending deceptive emails with the intent of gathering personal information, credentials, installing malware, or other malicious purposes
- Email is received that has one of three “threat types”: link, download, and attachment
- Users respond to them in a way informed by their folk model of the perceived threat:

Click a website link	Download from website	Open attachment
Toggle ad blocker	Patch their software	No Action

Table 5.1: Folk models and adherence to security advice

Computer security advice	Folk models			
	Graffiti	Burglar	Big Fish	Contractor
Use anti-virus software	∅	3	1	1
Keep anti-virus software updated	∅	∅	∅	1
Regularly scan computer with anti-virus	∅	∅	∅	1
Use security software (firewall, etc.)	2	2	2	1
Don't click on attachments	3	3	∅	∅
Be careful downloading from websites	2	2	1	1
Be careful which websites you visit	3	3	2	3
Disable scripting in web and mail	∅	∅	∅	1
Use good passwords	2	∅	2	1
Make regular backups	3	1	1	1
Keep patches up-to-date	3	3	1	1
Turn off computer when not in use	2	3	1	1

3: "It is very important to follow this advice."
2: "Following this advice might help, but it isn't all that important to do."
1: "It is not necessary to follow this advice."
∅: "This model has nothing to say about this advice, or there is insufficient data from interviews to determine an opinion."

Action	Folk models			
	Graffiti	Burglar	Big Fish	Contractor
Toggle ad blocker	Depends	Depends	Depends	No Change
Open attachment	Insecure	Insecure	Ambiguous	Ambiguous
Download from website	Depends	Depends	No Change	No Change
Click a website link	Insecure	Insecure	Depends	Insecure
Patch their software	Insecure	Insecure	No Change	No Change
No Action	No Change	No Change	No Change	No Change

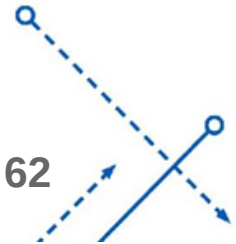


Table 5.1: Folk models and adherence to security advice

Computer security advice	Folk models			
	Graffiti	Burglar	Big Fish	Contractor
Use anti-virus software	∅	3	1	1
Keep anti-virus software updated	∅	∅	∅	1
Regularly scan computer with anti-virus	∅	∅	∅	1
Use security software (firewall, etc.)	2	2	2	1
Don't click on attachments	3	3	∅	∅
Be careful downloading from websites	2	2	1	1
Be careful which websites you visit	3	3	2	3
Disable scripting in web and mail	∅	∅	∅	1
Use good passwords	2	∅	2	1
Make regular backups	3	1	1	1
Keep patches up-to-date	3	3	1	1
Turn off computer when not in use	2	3	1	1

3: "It is very important to follow this advice."
2: "Following this advice might help, but it isn't all that important to do."
1: "It is not necessary to follow this advice."
∅: "This model has nothing to say about this advice, or there is insufficient data from interviews to determine an opinion."

Action	Folk models			
	Graffiti	Burglar	Big Fish	Contractor
Toggle ad blocker	Depends	Depends	Depends	No Change
Open attachment	Insecure	Insecure	Ambiguous	Ambiguous
Download from website	Depends	Depends	No Change	No Change
Click a website link	Insecure	Insecure	Depends	Insecure
Patch their software	Insecure	Insecure	No Change	No Change
No Action	No Change	No Change	No Change	No Change

IF siteContent = basic **AND** software = secure **THEN**
no change
ELSE
vulnerable
ENDIF

Results: False Vulnerability Mismatches

Table 5.2: Folk Model-Cross-Attack Verification Results: False Vulnerability Error States with Potential Ambiguity

	Link			Download			Attachment			No Action		
Folk Model	States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result
Graffiti	798	0.06	✘	816	0.06	✘	834	0.06	✘	930	0.06	✔
Burglar	798	0.07	✘	816	0.06	✘	834	0.06	✘	930	0.07	✔
Big Fish	816	0.07	✘	978	0.06	✔	858, 840	0.07, 0.07	✔ ●	1002	0.07	✔
Contractor	810	0.06	✘	978	0.08	✔	858, 846	0.06, 0.05	✔ ●	1002	0.06	✔

Note. A ✔ indicates a generic proof was returned, a ✘ indicates a generic counterexample was returned, and a ● indicates that an "ambiguity" counterexample was returned. Where ● is absent, an "ambiguity" proof was returned.

Results: False Safety Mismatches

Table 5.3: Folk Model-Cross-Attack Verification Results: False Safety Error States with Potential Ambiguity

	Link			Download			Attachment			No Action		
Folk Model	States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result
Graffiti	846	0.07	✔	834	0.06	✘	858	0.06	✔	906	0.07	✔
Burglar	846	0.06	✔	834	0.06	✘	858	0.06	✔	906	0.08	✔
Big Fish	864, 906	0.07, 0.06	✘ ●	930	0.06	✘	858, 870	0.06, 0.07	✔ ●	960	0.06	✔
Contractor	846	0.07	✔	954	0.07	✘	858, 870	0.06, 0.06	✔ ●	966	0.06	✔

Note. A ✔ indicates a generic proof was returned, a ✘ indicates a generic counterexample was returned, and a ● indicates that an "ambiguity" counterexample was returned. Where ● is absent, an "ambiguity" proof was returned.

Results: Blocking State Mismatches

Table 5.4: Folk Model-Cross-Attack Verification Results: Blocking States

	Link			Download			Attachment			No Action		
Folk Model	States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result
Graffiti	834	0.06	✔	834	0.06	✔	834	0.07	✔	834	0.06	✔
Burglar	834	0.07	✔	834	0.05	✔	834	0.07	✔	834	0.06	✔
Big Fish	834	0.07	✔	834	0.06	✔	834	0.06	✔	834	0.05	✔
Contractor	834	0.06	✔	834	0.07	✔	834	0.06	✔	834	0.07	✔

Note. A ✔ indicates a proof was returned, while a ✘ indicates a counterexample was returned.

Validation using real-world data

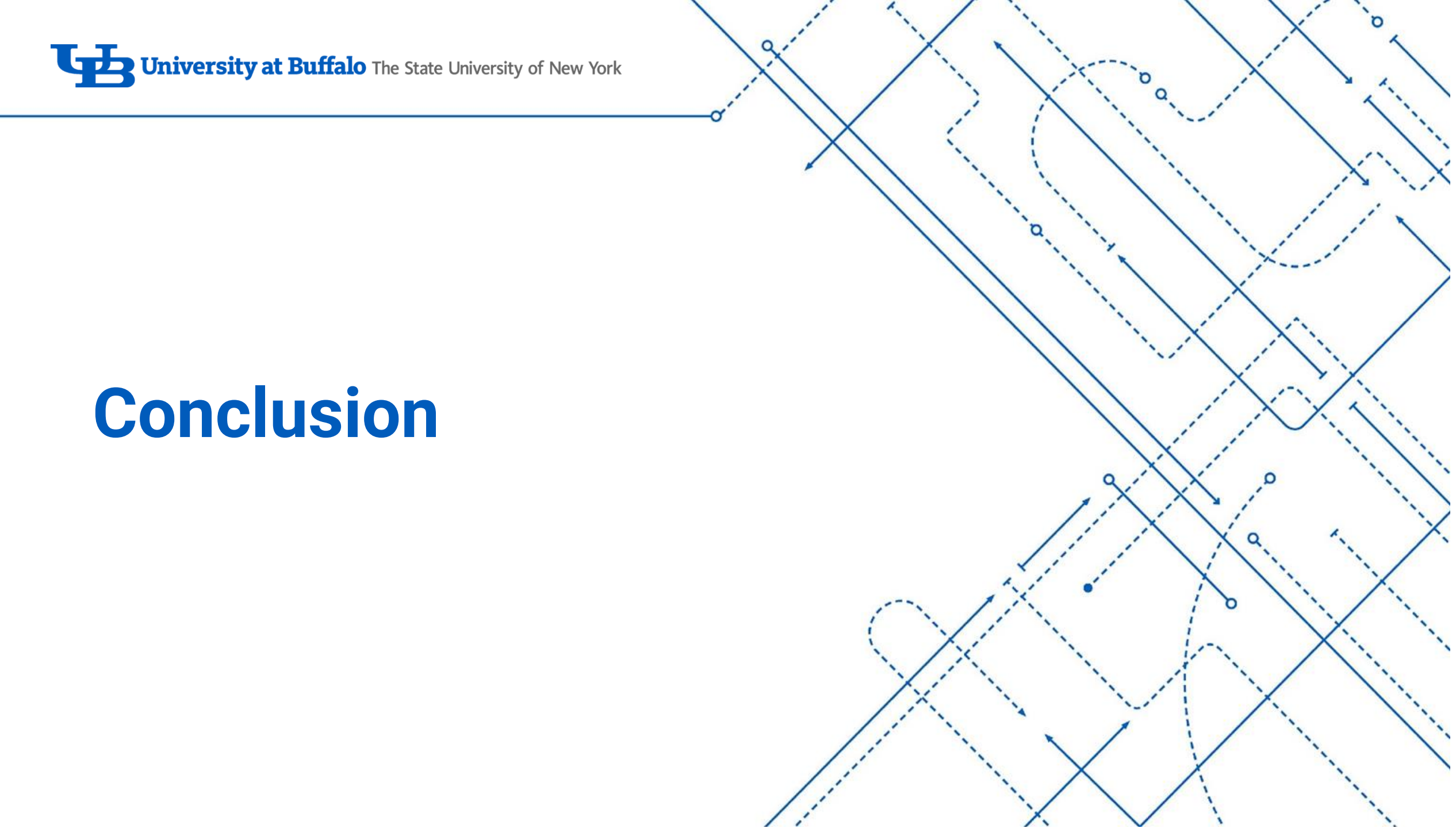
- Used data collected by Cofense, phishing training and simulation
  - 2016: 18mos, 40 million simulation emails, 1,000 organizations
  - 2017: 24mos, 52.4 million emails, 216,000 real emails, 1,400 organizations

- Significant agreement between data and findings
  - False security: Real users fell for link-based attacks more than others
  - False vulnerability: Real users drastically over-report emails with links as phishing attempts
  - High face validity associated with these findings

- Some limitations to using a “found” data set vs purpose-collected data

University at Buffalo The State University of New York

Conclusion



Significant findings and takeaways

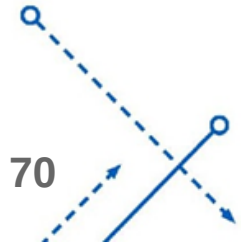
- A framework that …
  - extends Degani and Heymann (2002)'s concepts to cybersecurity
  - establishes formal analytic constructs for cybersecurity applications
  - improves rigor and brings runnability to folk modeling approaches

- To the best of our knowledge, these are completely novel developments

- Validated our findings using real-world data
  - Sometimes difficult to do with work in formal methods
  - Finds potential vulnerabilities that exist without altering code or breaking the law

Future work

- Additional CTL recovery specifications
- Update Wash (2010)'s folk modeling concepts for modern threats
- Expand AWS method to include complete bucket policies
- Investigate physical exploit delivery strategies (e.g., rubber duckies)
- Explore specific software utilities (Gmail vs Outlook vs Thunderbird)
- Integrate our method into cybersecurity testbeds (National Cyber Range)



Published and planned literature

Dissertation-related work:

Houser, A., and Bolton, M.L. (2017). Formal mental models for inclusive privacy and security. In *Proceedings of the Thirteenth Symposium on Usable Privacy and Security (SOUPS)*, Santa Clara CA.

Houser, A., Bolton, M.L., Bisantz, A.M., and Zhuang, J. (n.d.) Discovering unanticipated human-automation interaction in cybersecurity using formal methods techniques. In preparation.

Plus one more journal article and perhaps two more conference papers…

Additional work:

Houser, A., Ma, L., Feigh, K., and Bolton, M.L. (2017). Using formal methods to reason about taskload and resource conflicts in simulated air traffic scenarios. *Innovations in Systems and Software Engineering, 14*(1), pp. 1-14. DOI 10.1007/s11334-017-0305-2.

Bolton, M. L., Zheng, X., Molinaro, K., **Houser, A.**, and Li, M. (2016). Improving the scalability of formal human-automation interaction verification analyses that use task analytic models. *Innovation in Systems and Software Engineering, 13*(1), pp. 1-17. DOI 10.1007/s11334-016-0272-z.

Houser, A., Ma, L. M., Feigh, K., and Bolton, M. L. (2015). A formal approach to modeling and analyzing human taskload in simulated air traffic scenarios. In *Proceedings of the IEEE International Conference on Complex Systems Engineering*, 6 pages. Piscataway: IEEE.

Ma, L., **Houser, A.**, Feigh, K., and Bolton, M.L. (n.d.) An analysis of air traffic management concepts of operation using simulation and formal verification. Under review with the American Institute of Aeronautics and Astronautics (AIAA).

Published and planned literature

Published articles:

Houser, A., Ma, L., Feigh, K., and Bolton, M.L. (2017). Using formal methods to reason about taskload and resource conflicts in simulated air traffic scenarios. *Innovations in Systems and Software Engineering, 14*(1), pp. 1-14. DOI 10.1007/s11334-017-0305-2.

Bolton, M. L., Zheng, X., Molinaro, K., **Houser, A.**, and Li, M. (2016). Improving the scalability of formal human-automation interaction verification analyses that use task analytic models. *Innovation in Systems and Software Engineering, 13*(1), pp. 1-17. DOI 10.1007/s11334-016-0272-z.

Articles under review:

Ma, L., **Houser, A.**, Feigh, K., and Bolton, M.L. (n.d.) An analysis of air traffic management concepts of operation using simulation and formal verification. Under review with the American Institute of Aeronautics and Astronautics (AIAA).

Articles in preparation:

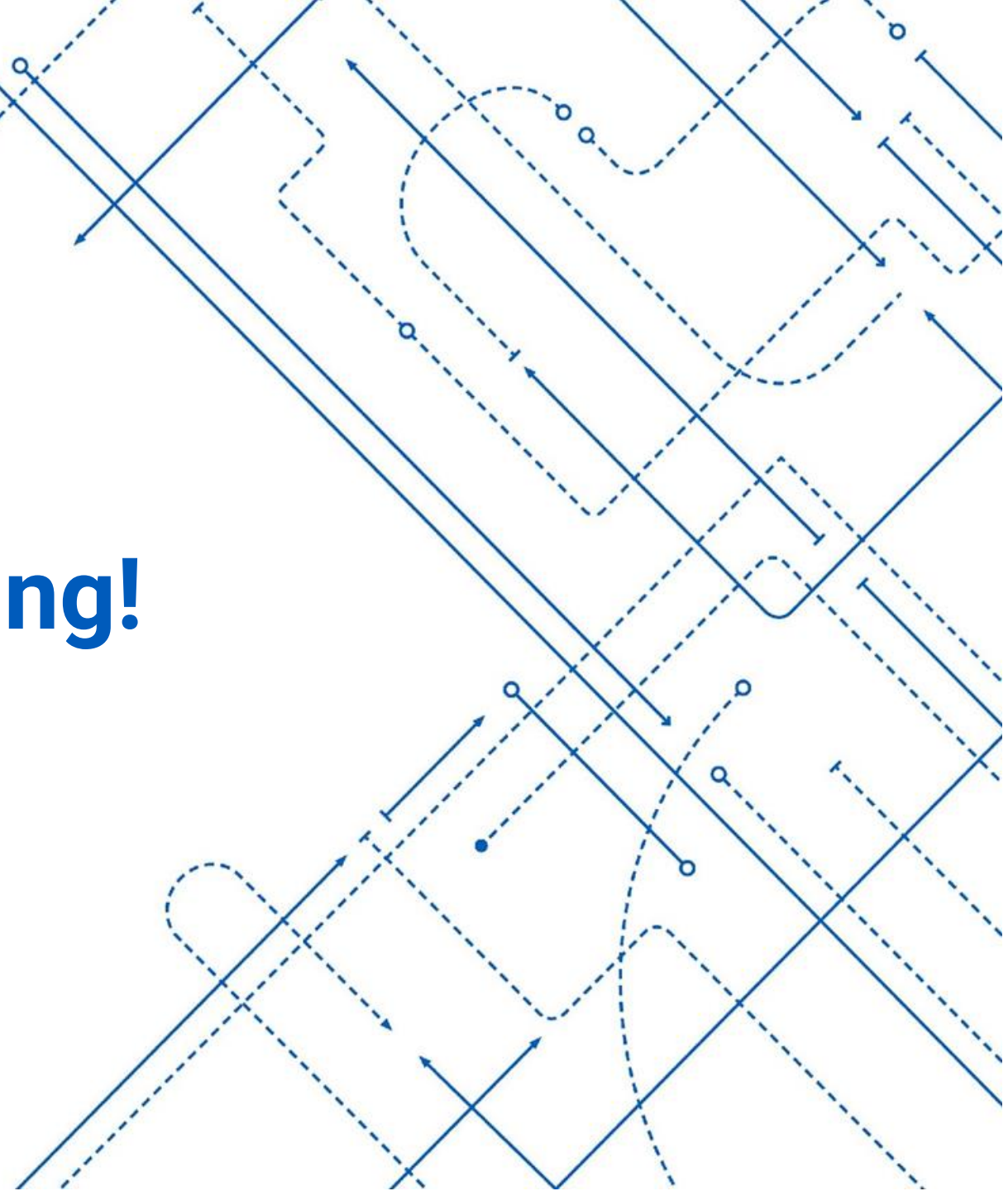
Houser, A., Bolton, M.L., Bisantz, A.M., and Zhuang, J. (n.d.) Discovering unanticipated human-automation interaction in cybersecurity using formal methods techniques.

Published conference papers:

Houser, A., and Bolton, M.L. (2017). Formal mental models for inclusive privacy and security. In *Proceedings of the Thirteenth Symposium on Usable Privacy and Security (SOUPS)*, Santa Clara CA.

Houser, A., Ma, L. M., Feigh, K., and Bolton, M. L. (2015). A formal approach to modeling and analyzing human taskload in simulated air traffic scenarios. In *Proceedings of the IEEE International Conference on Complex Systems Engineering*, 6 pages. Piscataway: IEEE.

University at Buffalo The State University of New York



Thank you for attending!

adam.m.houser@gmail.com

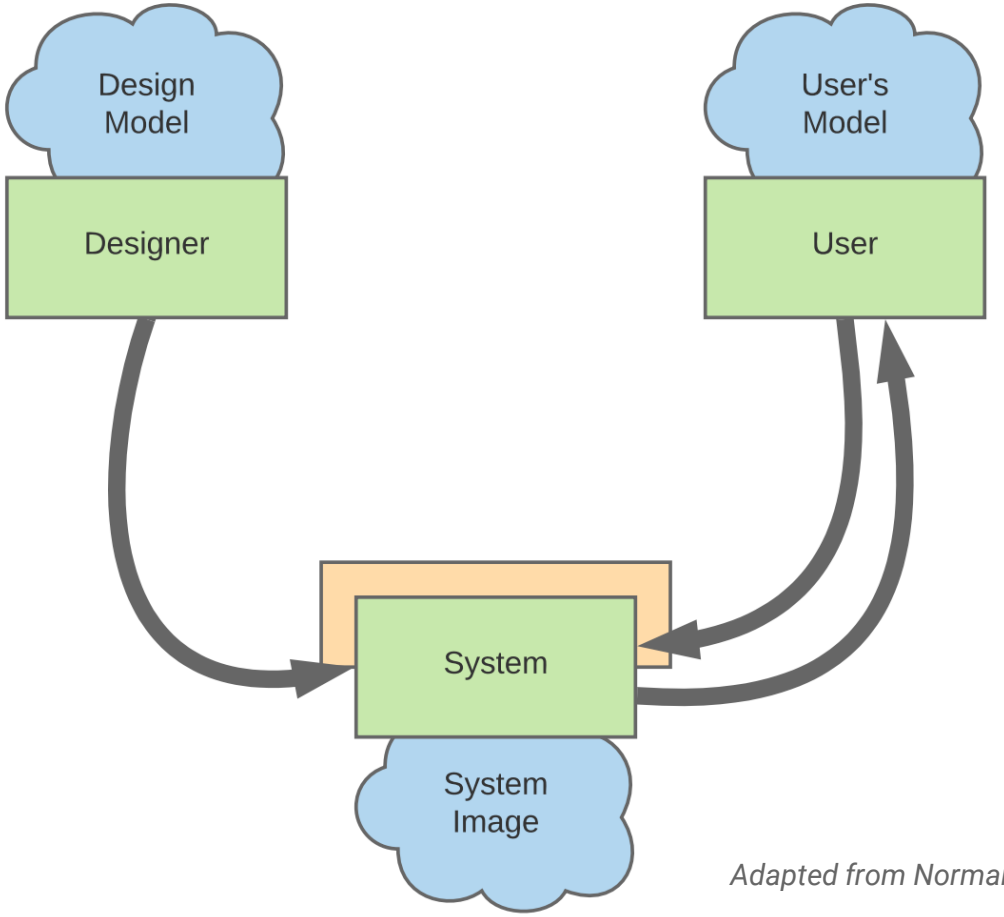
@neutrinos4all

appliedcaffeine.org

University at Buffalo The State University of New York

Reserve Slides

Norman on mental models

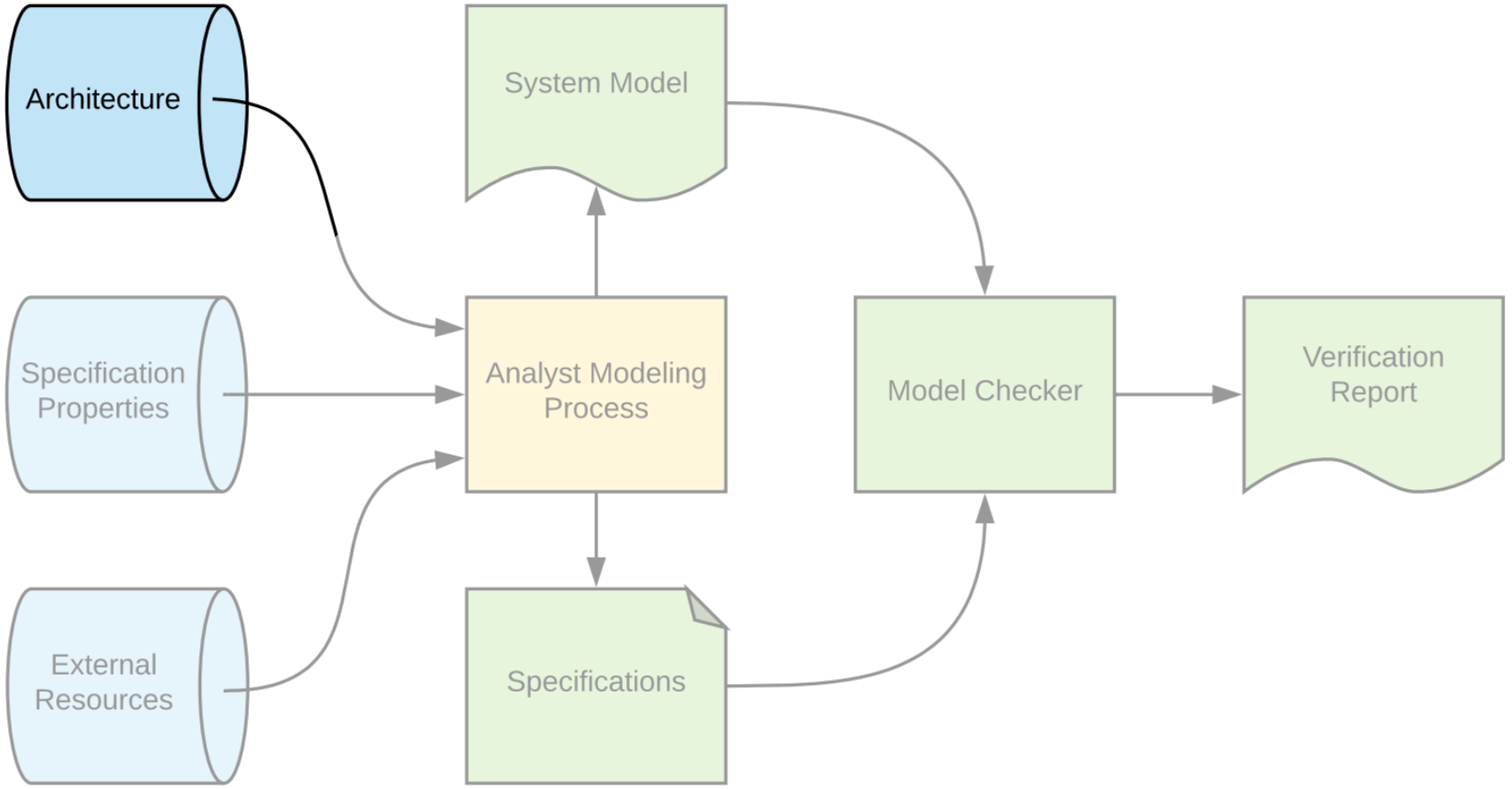


Adapted from Norman (1983, 2007)

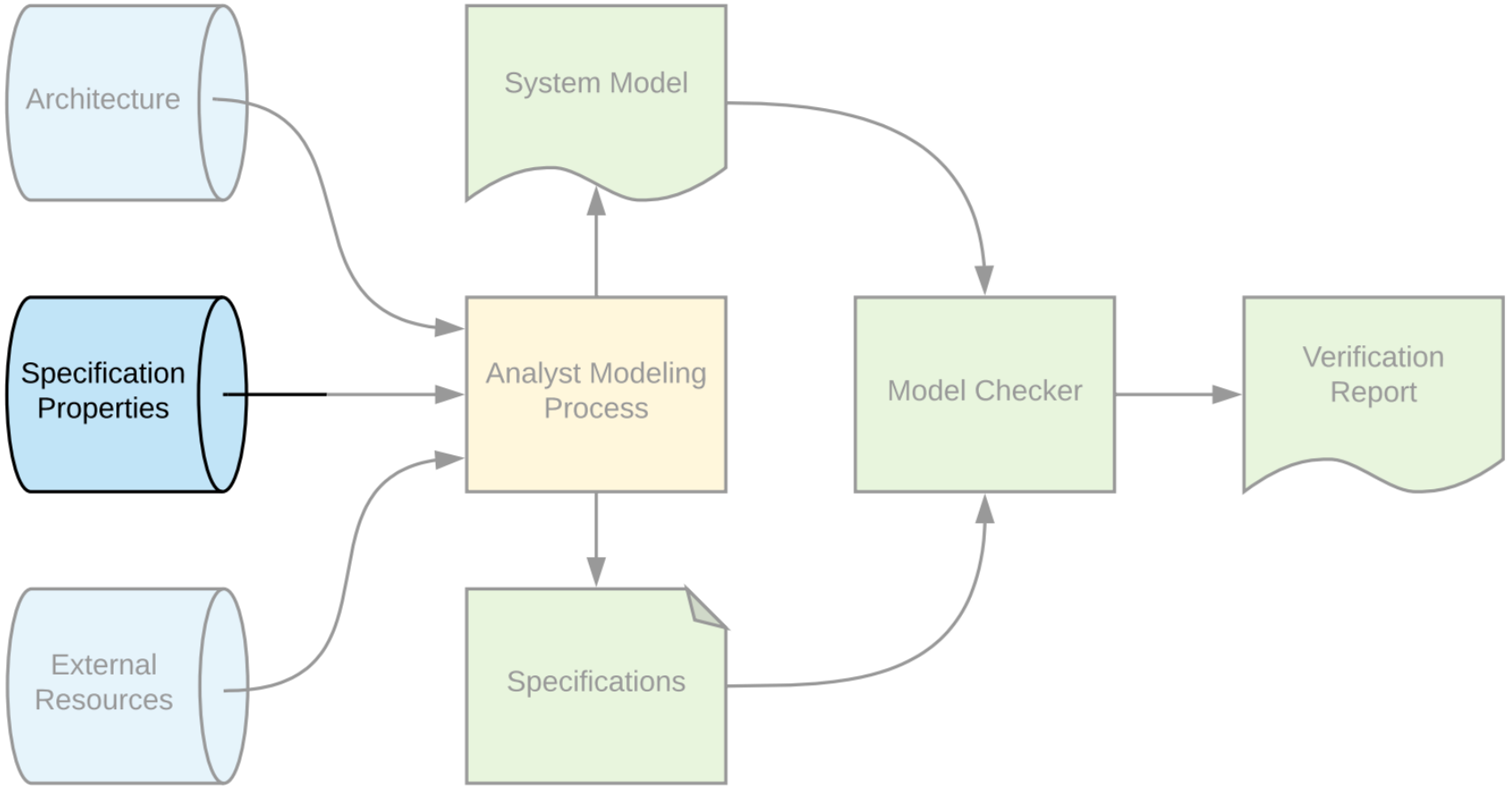
University at Buffalo The State University of New York

Generic formulation: Stepwise overview

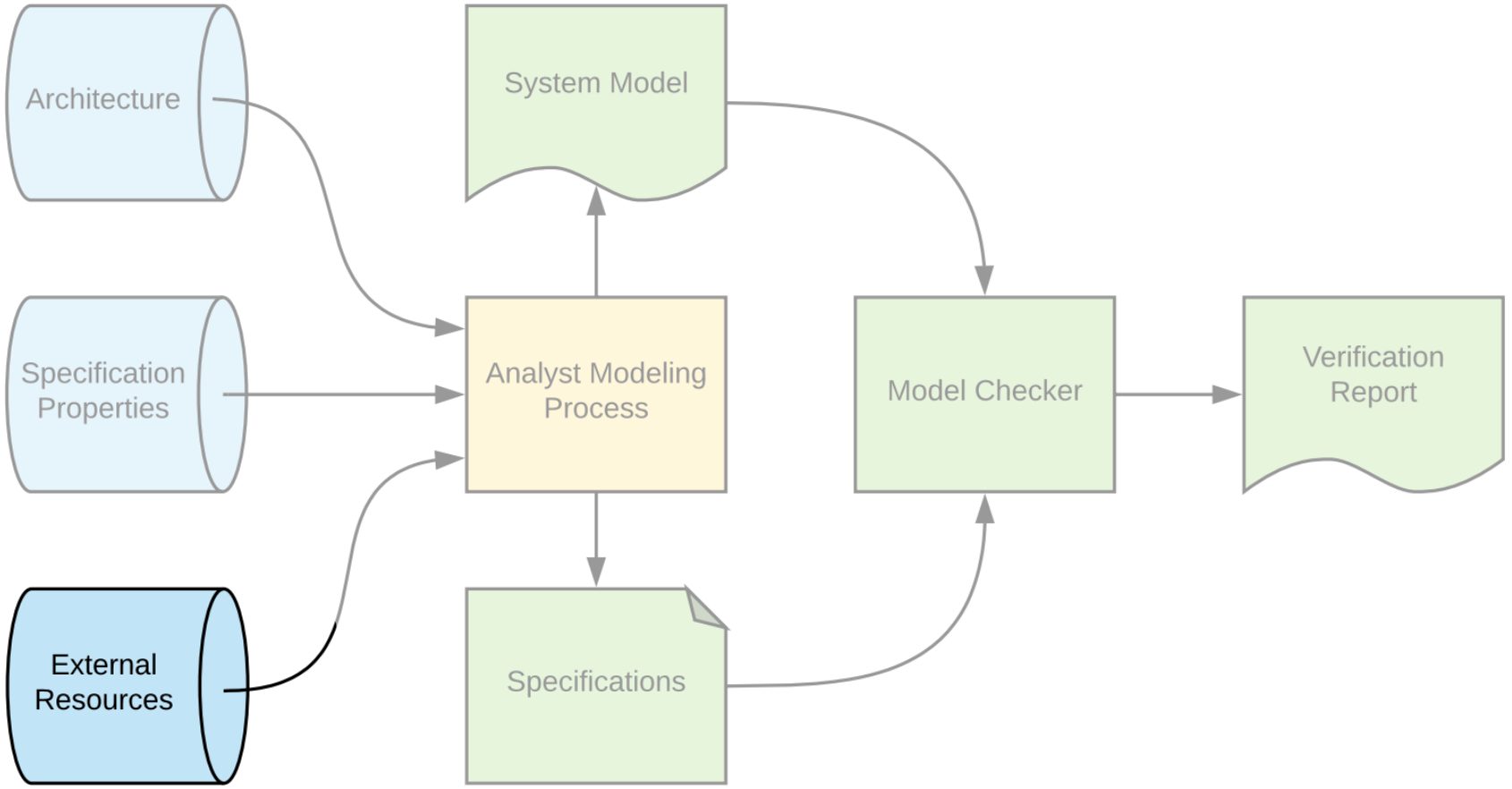
Generic formulation of the method



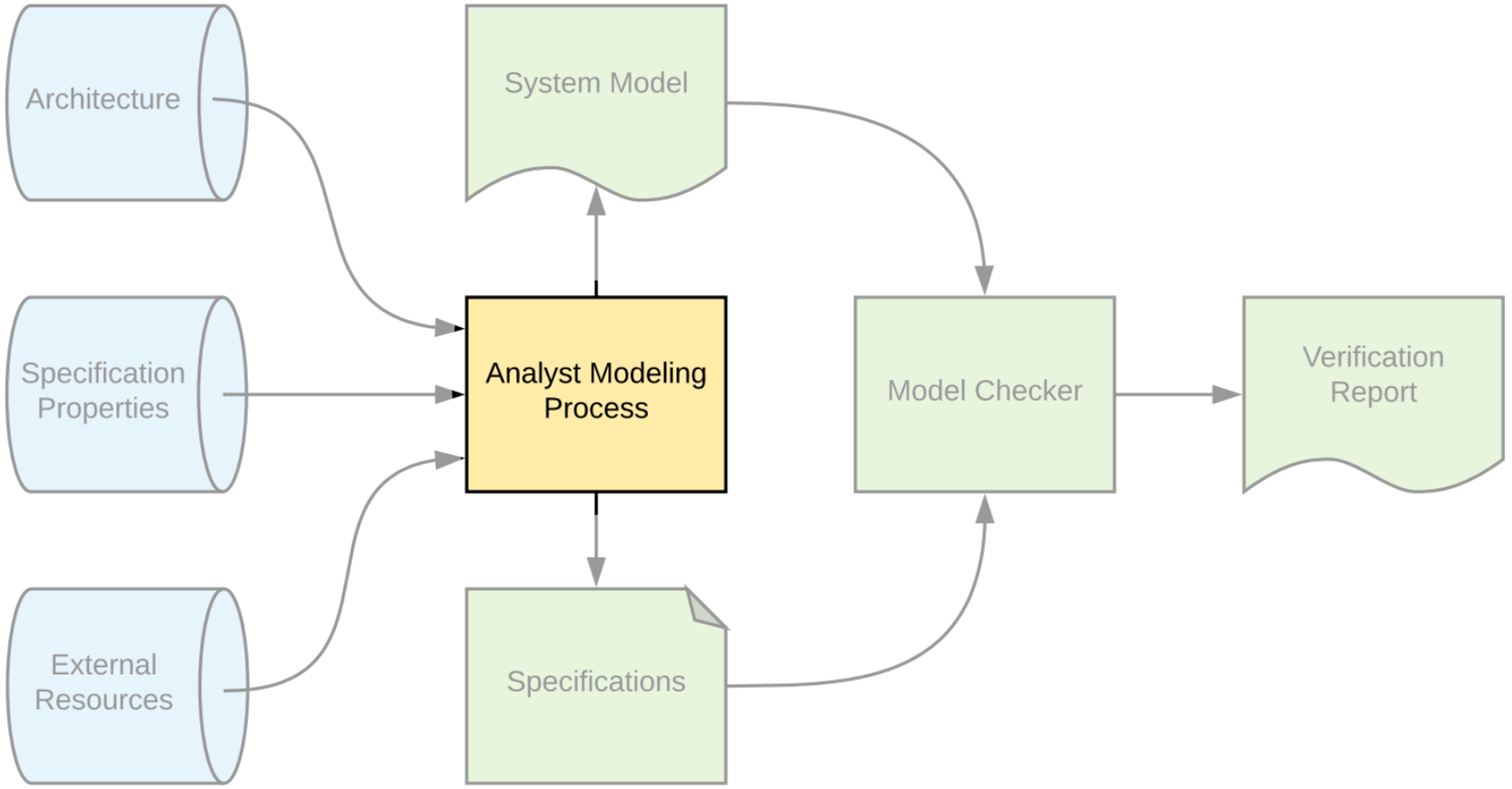
Generic formulation of the method



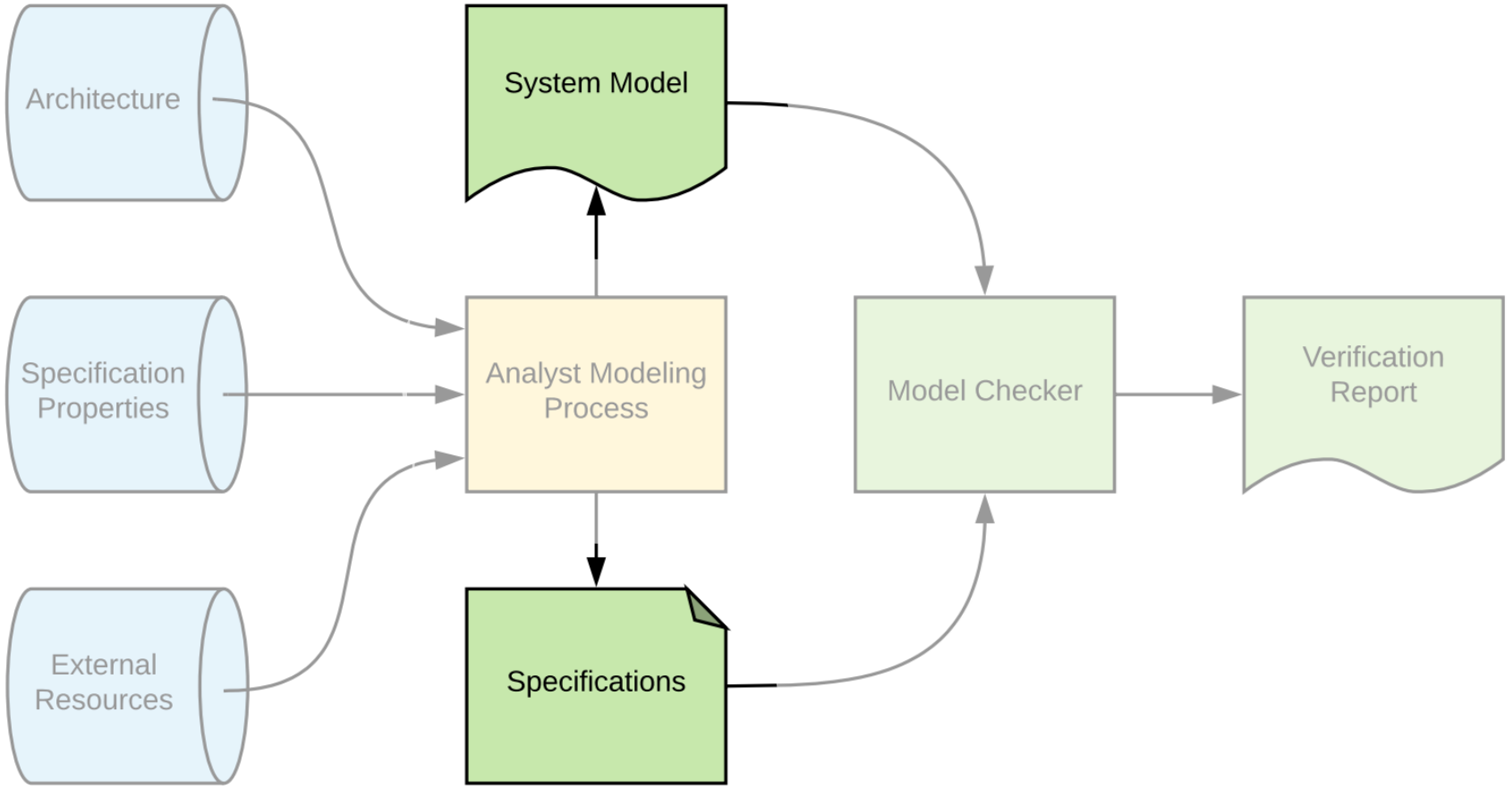
Generic formulation of the method



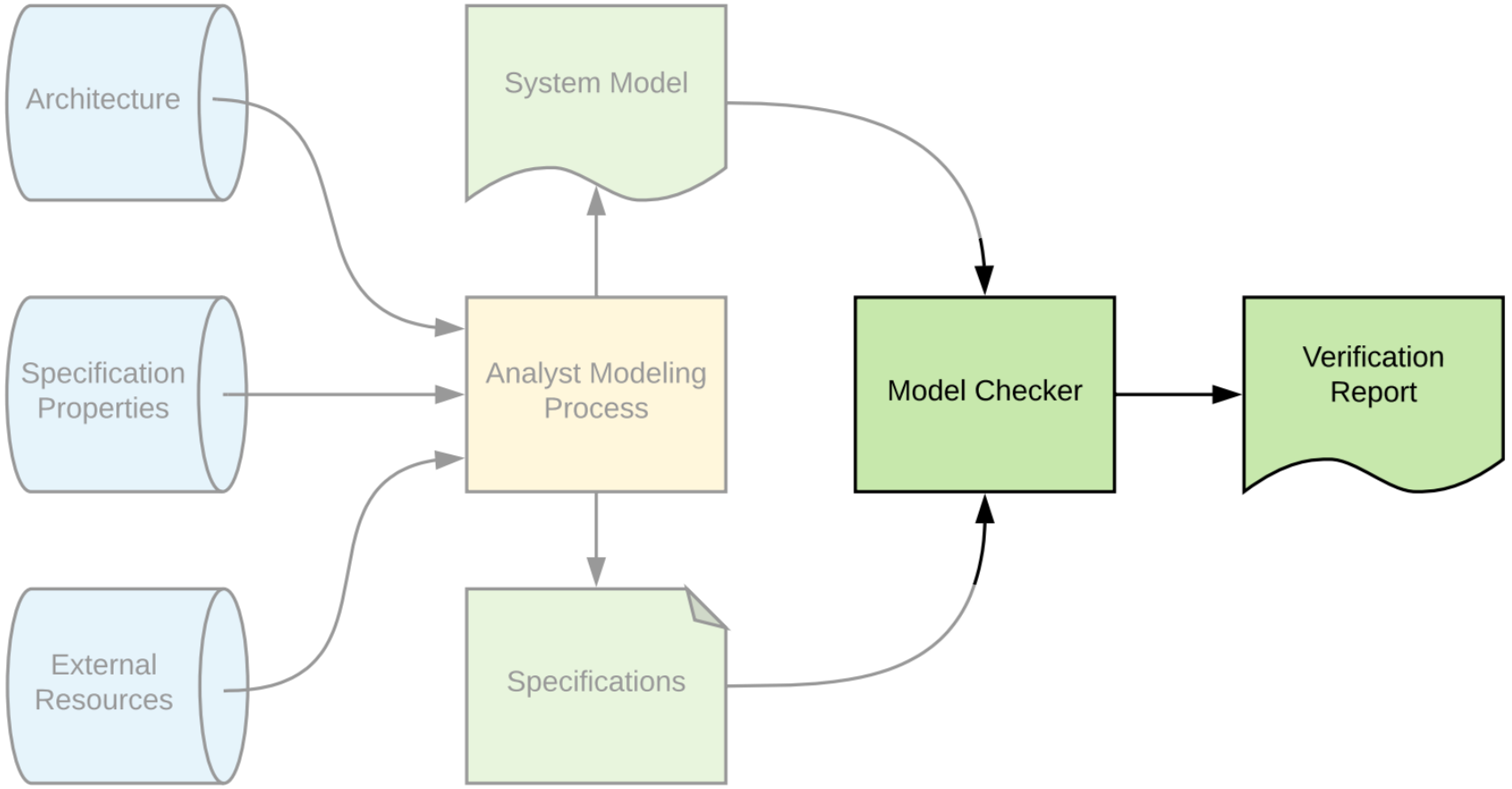
Generic formulation of the method



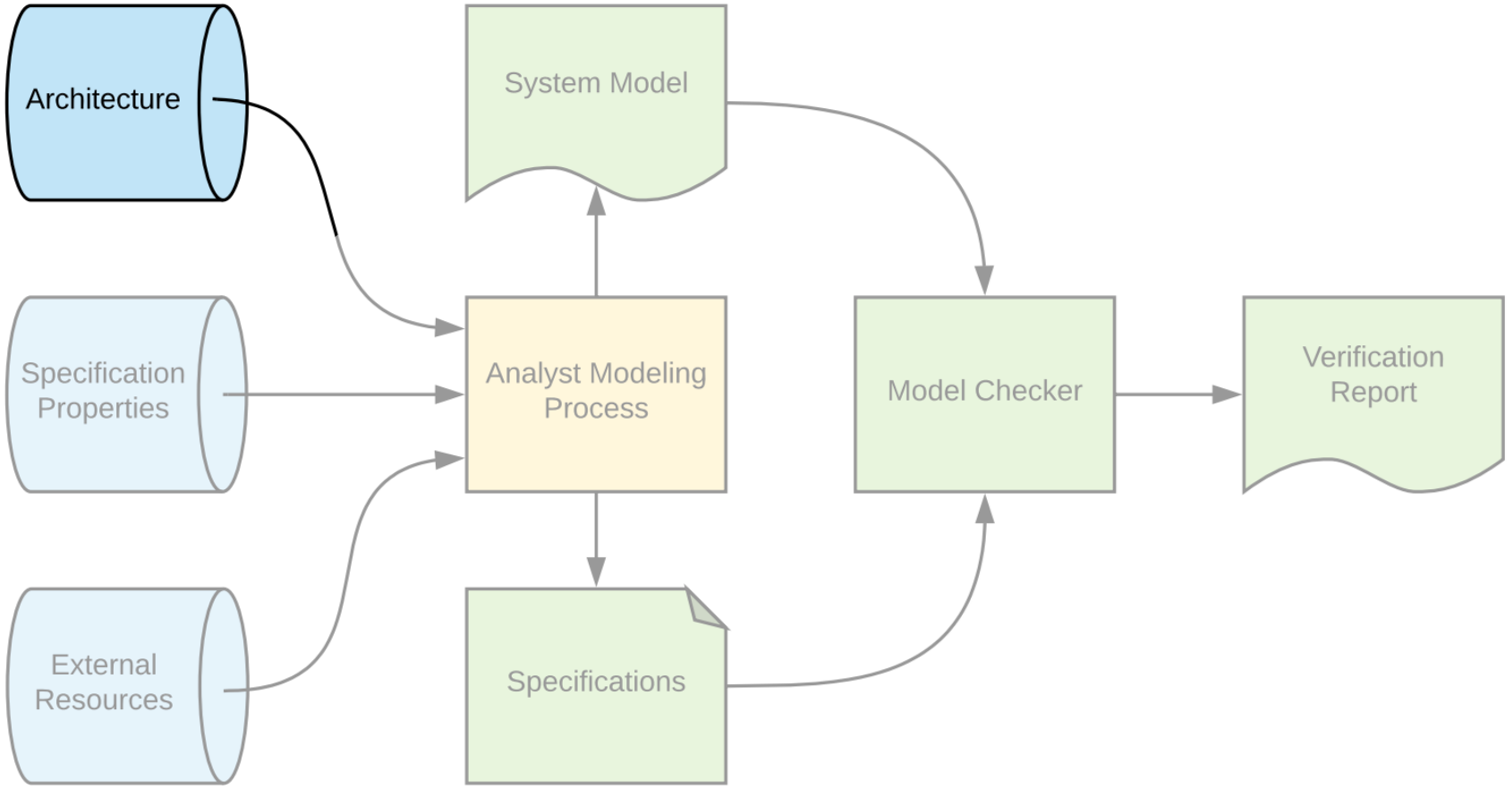
Generic formulation of the method



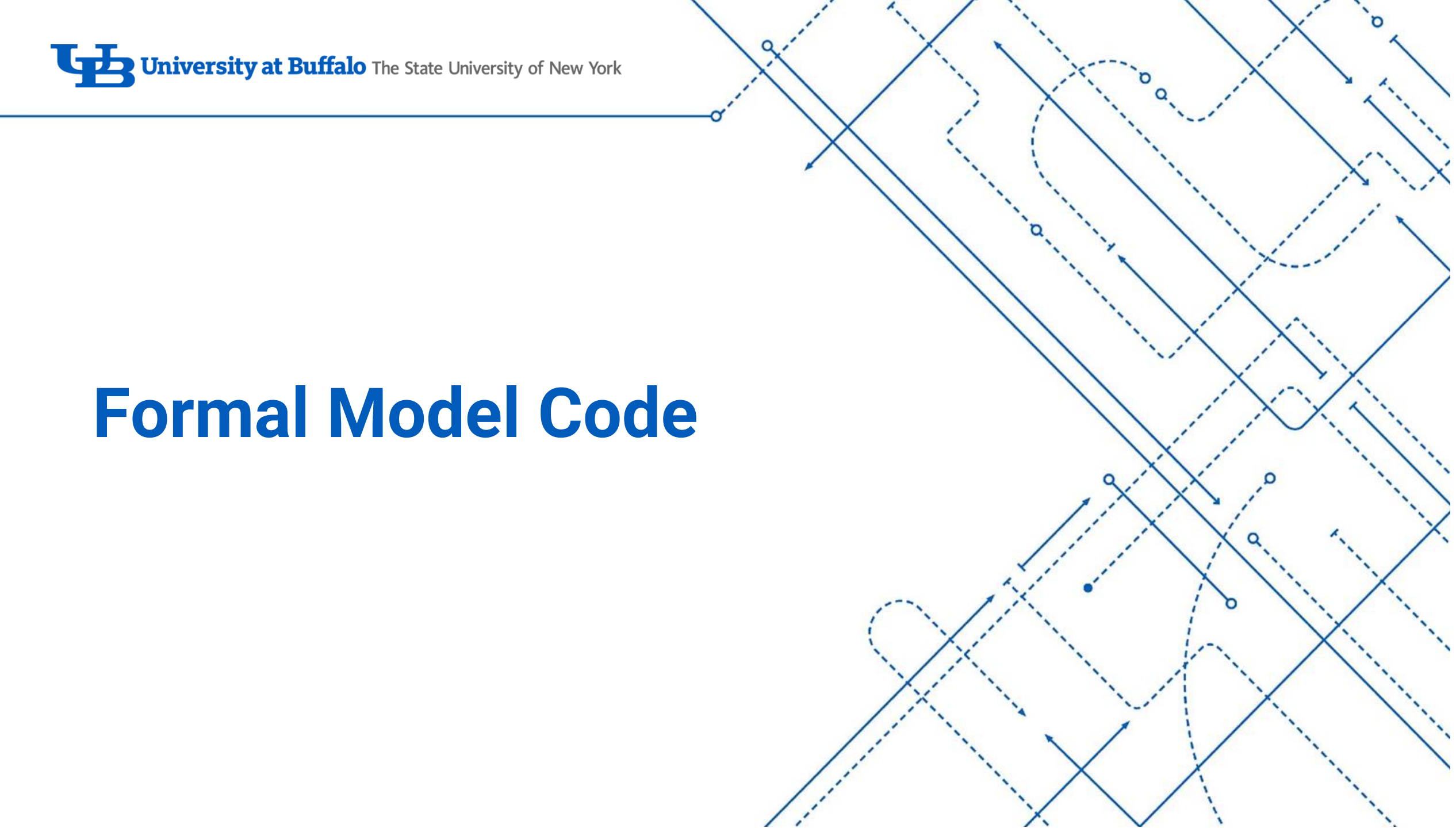
Generic formulation of the method



Method Architecture



University at Buffalo The State University of New York



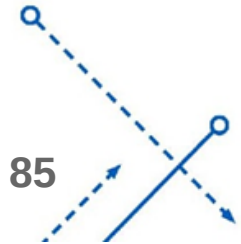
Formal Model Code

The Action Generator module

```
actionPerm IN IF action = change_ObjectPermissions OR action = create_Bucket OR action =
        change_BucketPermissions
                THEN {x : permissionSetting | NOT (x.public AND isPrivate?(x))}
                ELSE {x : permissionSetting | (NOT x.public) AND isPrivate?(x)}
                ENDIF;
```

```
isPublic?(perm : permissionSetting) : BOOLEAN  = perm.public AND (perm.read OR perm.write OR
        perm.readPermissions OR perm.writePermissions);
```

```
isPrivate?(perm : permissionSetting) : BOOLEAN = NOT perm.read AND NOT perm.write AND NOT
        perm.readPermissions AND NOT perm.writePermissions;
```



The System and User modules: Example 1

```
[ ] action = create_Object AND bucketExists AND NOT objectExists -->
        objectExists'                   =  TRUE;
        objectEncryption'               =  IF actionEnc = none_Option
                                                THEN bucketEncryption
                                                ELSE actionEnc
                                                ENDIF;
        objectACL'                      =  actionPerm;


[ ] action = create_Object AND thinkBucketExists AND NOT thinkObjectExists -->
        thinkObjectExists'              =  TRUE;
        thinkObjectEncryption‘          =  actionEnc;
        thinkObjectACL'                 =  actionPerm;
```

The System and User modules: Example 2

softwareRead = IF objectExists AND objectACL.public AND (objectACL.read OR objectACL.readPermissions OR objectACL.writePermissions)
THEN insecure
ELSE secure
ENDIF;

userRead = IF thinkObjectExists AND thinkObjectACL.public AND (thinkObjectACL.read OR thinkObjectACL.readPermissions) **AND thinkObjectEncryption = none_Option**
THEN vulnerable
ELSE safe
ENDIF;

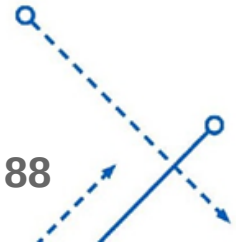
The Comparator module

falseVulnerabilityReadMM = (userRead = vulnerable AND softwareRead = secure);

falseVulnerabilityWriteMM = (userWrite = vulnerable AND softwareWrite = secure);

falseSafetyReadMM = (userRead = safe AND softwareRead = insecure);

falseSafetyWriteMM = (userWrite = safe AND softwareWrite = insecure);



Limitations

- Specification-based model checking finds one path to failure; could be others
- Investigated method of formulating specifications tailored to find scenarios in results

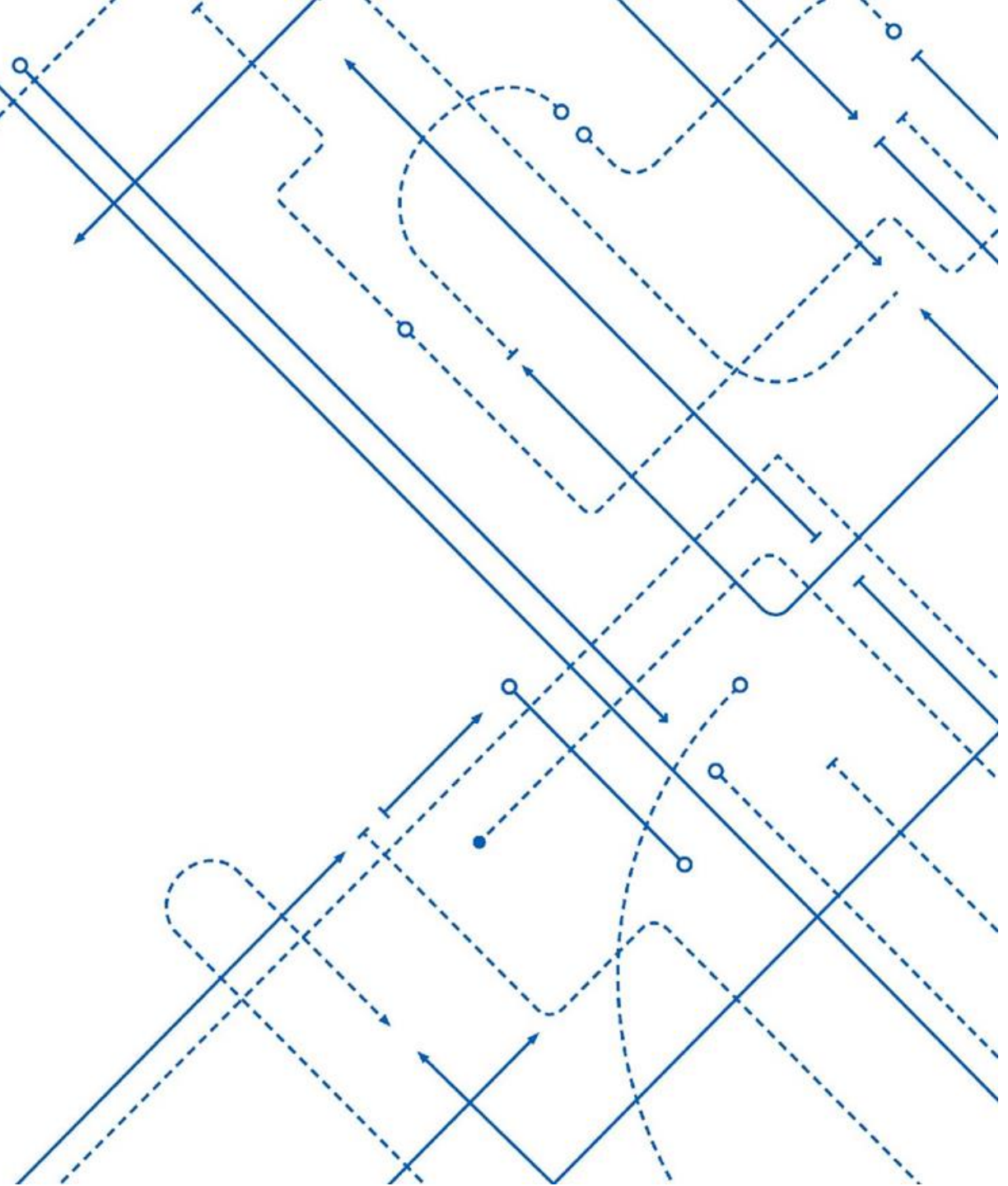
Table 4.2: Amazon AWS S3 Use Case Findings Validation Results

Authenticated User Mismatch			Encrypted but Readable			Surprising Inheritance		
States	Time (s)	Result	States	Time (s)	Result	States	Time (s)	Result
3,777,972	1.19	✘	3,833,412	0.14	✘	2,200,836	0.38	✘

Note. A ✔ indicates a proof was returned, while a ✘ indicates a counterexample was returned.

University at Buffalo The State University of New York

Phishing Model Validation



Validation using real-world data

Table 5.5: Cofense Phishing Simulation Responses by Attack Strategy

Template	Click-only	Data entry	Attachment-based
File from scanner	31.1		
Package delivery—alternate	25.8		
Unauthorized access		24.8	
Digital fax	20.2		
Package delivery	19.7		
Scanned file			18.4
RSA phish	13.1		
Password survey		12.9	
Secure email			12.2
Adobe security update	7.0		
Restaurant gift receipt			6.7
Google docs link	6.0		
Funny pictures			3.4
Relative ratios	.610	.187	.202

Note. The values reported here indicate the percentage of users that clicked on the email out of all users who received it. For example, 31.1% of all users receiving the "file from scanner" phishing email clicked on it.

Validation using real-world data

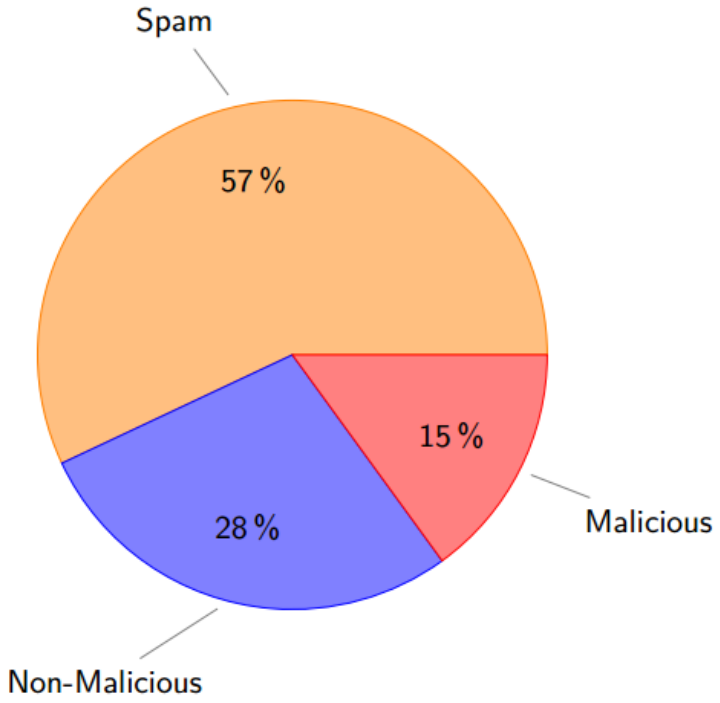


Figure 5.5: All emails, reported malicious versus non-malicious.

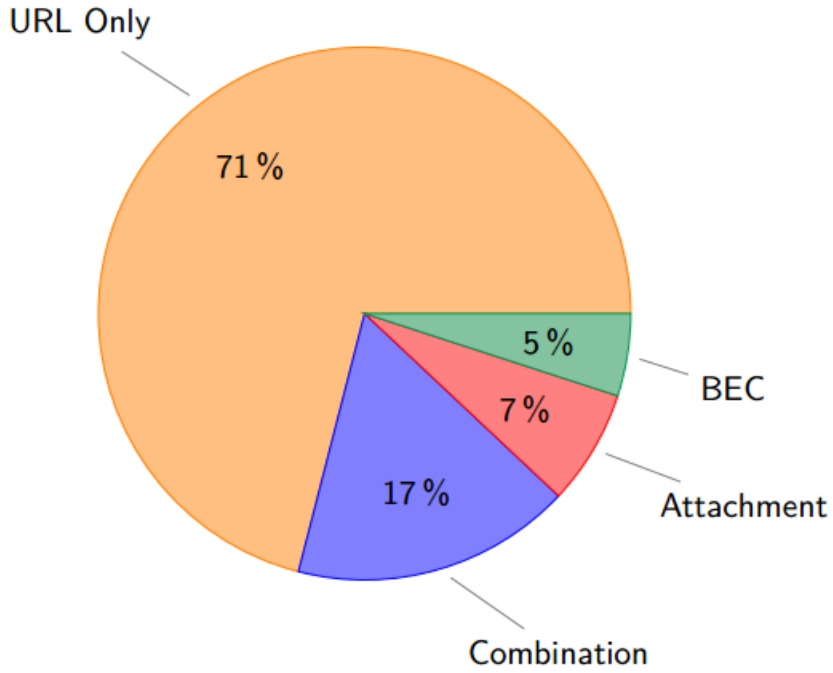


Figure 5.6: Confirmed malicious emails by attack type.

References

Ashok, I. (2017, December 1). National credit federation data leak: Over 100gb of sensitive customer data was left exposed online. International Business Times. Retrieved from http://www.ibtimes.co.uk/national-credit-federation-data-leak-over-100gb-sensitive-customer-data-was-left-exposed-online-1649709

Baier, C., & Katoen, J. P. (2008). Principles of model checking. MIT press.

Blythe, J., & Camp, L. J. (2012). Implementing mental models. In 2012 IEEE Symposium on Security and Privacy Workshops (SPW) (p. 86-90).

Bolton, M. L., Bass, E. J., & Siminiceanu, R. I. (2013). Using formal verification to evaluate human-automation interaction: A review. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43(3), 488-503.

Cameron, D. (2017, May 31). Top defense contractor left sensitive pentagon files on amazon server with no password. Gizmodo. Retrieved from https://gizmodo.com/top-defense-contractor-left-sensitive-pentagon-files-on-1795669632

Camp, L. J. (2004). Mental models of computer security. Financial Cryptography, 3110, 106-111.

Camp, L. J. (2009). Mental models of privacy and security. IEEE Technology and Society Magazine, 28(3), 37-46.

Cofense. (2016). Enterprise Phishing Susceptibility and Resiliency Report (Tech. Rep.). PhishMe. Retrieved from https://cofense.com/enterprise-phishing-susceptibility-report/

Cofense. (2017). Enterprise Phishing Susceptibility and Defense Report (Tech. Rep.). PhishMe. Retrieved from https://cofense.com/whitepaper/enterprise-phishing-resiliency-and-defense-report/

Collett, S. (2014, May 21). Five new threats to your mobile device security. CSO Online. Retrieved from http://www.csoonline.com/article/2157785/data-protection/five-new-threats-to-your-mobile-device-security.html

Craik, K. (1943). The Nature of Explanation. Cambridge University Press.

D'Andre, R. (1987). A folk model of the mind. In Cultural Models in Language and Thought (p. 112-148).

Degani, A., Shafto, M., & Kirlik, A. (1999). Modes in human-machine systems: Constructs, representation, and classification. The International Journal of Aviation Psychology, 9(2), 125-138.

Degani, A., & Heymann, M. (2002). Formal verification of human-automation interaction. Human Factors, 44(1), 28-43.

References

de Kleer, J., & Brown, J. S. (1981). Mental models of physical mechanisms and their acquisition. In Cognitive Skills and their Acquisition (p. 285-309). Erlbaum: Hillsdale, NJ.

Dekker, S., & Hollnagel, E. (2004). Human factors and folk models. Cognition, Technology, and Work, 6, 79-86.

FBI: Cyber Crime. (2016). Retrieved 15 February 2017, from https://www.fbi.gov/investigate/cyber

Gartner. (2015, Sep 23). Gartner says worldwide information security spending will grow almost 4.7 percent to reach $75.4 billion in 2015 (Tech. Rep.). Retrieved from https://www.gartner.com/newsroom/id/3135617

Juniper Research. (2015, May 12). Cybercrime will cost businesses over $2 trillion by 2019 (Tech. Rep.). Retrieved from https://www.juniperresearch.com/press/press-releases/cybercrime-cost-businesses-over-2trillion

Kumaraguru, P., Cranshaw, J., Acquisti, A., Cranor, L., Hong, J., Blair, M. A., & Pham, T. (2009). School of phish: a real-world evaluation of anti-phishing training. In Proceedings of the 5th symposium on usable privacy and security (p. 3).

Morgan, S. (2016, Mar 9). Worldwide cybersecurity spending increasing to $170 billion by 2020. Forbes. Retrieved from http://www.forbes.com/sites/stevemorgan/2016/03/09/worldwide-cybersecurity-spending-increasing-to-170-billion-by-2020/#7cde23d576f8

Newman, L. H. (2017, June 19). The scarily common screw-up that exposed 198 million voter records. Wired. Retrieved from https://www.wired.com/story/voter-records-exposed-database/

Norman, D. A. (1983). Some observations on mental models. In Mental Models (p. 7-14). Erlbaum: Hillsdale, NJ.

Norman, D. A. (2013). The design of everyday things: Revised and expanded edition. Basic Books.

Oishi, M., Hwang, I., & Tomlin, C. (2003). Immediate observability of discrete event systems with application to user-interface design. In Proceedings of the 42nd IEEE Conference on Decision and Control, 2003 (Vol. 3, p. 2665-2672).

O'Sullivan, D. (2017, November 28). Black box, red disk: How top secret nsa and army data leaked online. UpGuard. Retrieved from https://www.upguard.com/breaches/cloud-leak-inscom

References

Sheng, S., Holbrook, M., Kumaraguru, P., Cranor, L. F., & Downs, J. (2010). Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In Proceedings of the SIGCHI conference on human factors in computing systems (p. 373-382).

Siadati, H., Palka, S., Siegel, A., & McCoy, D. (2017). Measuring the effectiveness of embedded phishing exercises. In 10th USENIX workshop on cyber security experimentation and test (CSET 17).

Wash, R. (2010). Folk models of home computer security. In Proceedings of the Sixth Symposium on Usable Privacy and Security (SOUPS) 2010 (p. 11).

Whittaker, Z. (2017, July 12). Millions of verizon customer records exposed in security lapse. InfoSecurity Magazine. Retrieved from http://www.zdnet.com/article/millions-verizon-customer-records-israeli-data/