SynBioTrace: Integrating Safety and Security Artifacts to Build Assurance Cases for Synthetic Biology Applications

Justin Firestone
Dept. of Computer Science & Engineering
University of Nebraska–Lincoln
Lincoln, NE 68588
justin.firestone@unl.edu

Myra B. Cohen, Robyn R. Lutz
Dept. of Computer Science
Iowa State University
Ames, IA 50011
{mcohen,rlutz}@iastate.edu

***Abstract*—The rapidly advancing cyber-physical domain of synthetic biology modifies the functionality of micro-organisms which can act as living computational devices. Applications include intelligent drug delivery, customized cancer therapies, and pollution detection and mitigation. While many synthetic biology applications have been proposed, prototyped, and even deployed, these systems often lack standard approaches to verify their safety and security. One approach, the assurance case, provides evidence demonstrating proper implementation in the target application, and is often used in other safety-critical domains. However, synthetic biologists lack guidance in developing assurance arguments and tracing safety and security requirements to evidence as required for building assurance cases. Although there has been some research combining safety and security artifacts, such techniques often require extensive expertise from different domains and may not be accessible to synthetic biologists. In this paper we propose SynBioTrace, an assistive process to help propagate information from risk-based analyses of such systems to preliminary assurance cases. SynBioTrace preserves traceability among its steps so that the assurance case can be further refined. We apply and evaluate it through a case study based on a real-world synthetic biology application. Our case study suggests this approach could aid synthetic biologists in identifying, documenting, and structuring safety and security artifacts, as well as linking evidence to support traceability for a complete, integrated assurance case.**

***Index Terms*—Assurance Cases, Synthetic Biology, Safety & Security Evidence**

I. Introduction

Synthetic biology is the practice of engineering living organisms by modifying their DNA. The discipline has progressed rapidly over the last 30 years due to advances that provided solutions for key technical challenges [1]. Synthetic biology has been defined as a process "to design new, or modify existing, organisms to produce biological systems with new or enhanced functionality according to quantifiable design criteria." Successful projects include engineering bacteria to produce synthetic biofuels, to sense and mitigate pollution, to organize and generate body tissues, to perform medical diagnostics, and, increasingly, to perform sophisticated computations [2].

Synthetic biologists design new functionality, encode it into DNA strands, and insert those DNA strands into living organisms such as the common (non-toxic) K-12 strain of *Escherichia coli* (*E. coli*). The engineered organisms replicate this new DNA along with its native DNA to build proteins that perform designed functionalities. It is possible to encode different types of logic within cells using different DNA sequences [3]. Engineers design bacteria which, similar to silicon devices, can sense, actuate, compute, and communicate based on the flow of molecular signals at the nanoscale. These new organisms, and the focus of this paper, can be described as living computational systems or even *biological software*.

Biological software, like traditional software, must be tested, validated, and verified for correctness and dependability within its run-time environment. Unlike many traditional silicon-based systems, engineered bacteria can interact with other life and with the environment while performing computational functions. Hence many of these applications are safety-critical, and likely more are safety critical than those previously considered to be, such as medical diagnostics and therapeutics [4].

Given the rapid growth in quantity and quality of synthetic biology applications in a wide variety of safety-critical domains, there has been increasing attention on the need for regulation and certification of biological software [5]–[7]. However, synthetic biologists are unlikely to have expertise in software testing and safety assurance, and if we are to expect certification of the safety and security of their biological software, we should assist them and all other stakeholders by providing initial guidance on how to generate meaningful, suitable artifacts. Research has suggested assurance cases [8]–[10], can be built for synthetic biology as a way to facilitate regulatory oversight [11]–[13].

This paper focuses on two current challenges for building synthetic biology assurance cases: (1) synthetic biologists lack guidance for creating assurance cases; and (2) any proposed solution needs to address safety together with security to support of traceability of evidence. To overcome these challenges, we are inspired by the safety community and build on prior work regarding traceable links between safety and security artifacts [14]–[16]. Fault trees and attack-defense trees are often the first artifacts regulators request. We show how to integrate information from these artifacts to incrementally build an initial assurance case that provides traceability among goals, strategies, assumptions, contexts, and evidence. We call the process of generating these artifacts and their traceable relationships across domains *SynBioTrace*. The main contributions of this work are:

1) A hazards-based, assistive process to help develop and refine assurance cases for synthetic biology applications, and
2) A case study of a synthetic biology kill-switch to demonstrate SynBioTrace's potential.

In the next section we present some background and a motivating example. In Section III we provide an overview of SynBioTrace. We apply and evaluate it on a case study in Section IV and discuss the results in Section V. Section VI describes related work. We end with conclusions and future work.

II. Motivating Example

We present a motivating example based on a temperature-sensitive kill-switch from a synthetic biology application to inhibit methane production in cattle (methanogenesis). This kill-switch was a team project entered in an international synthetic biology competition (iGEM) which has over 400 team entries and thousands of attendees per year [17]. (See Section IV for a more complete description of the kill-switch.) Recently, some winning iGEM teams have created partial assurance cases for their synthetic biology applications. However, guidance on how to develop an assurance case for a synthetic biology system is lacking. In our example application, the high-level goal is to engineer bacteria that lives in a cattle's rumen and reduces its methane output. This has both environmental and economic benefits: it reduces the amount of methane released into the atmosphere and lowers costs by increasing feed efficiency. However, to be released into the environment, it needs to be trusted and safe.

The complete methane-inhibition system requires adding engineered bacteria to cattle feed so the bacteria become part of the biome in digestive systems. The bacteria then interact with the methane metabolic pathway to reduce the production of methane [17]. To prevent the engineered bacteria from interacting with the external environment, the team added a safety requirement that the bacteria are only able to live and operate within the digestive system of the cattle.

Biological kill-switches are defined as "artificial systems that result in cell death under certain conditions" [18], and are an important safety mechanism built into many synthetic biology applications. A kill-switch ensures engineered organisms will not grow out of control after their necessary functions have been performed, limiting their opportunity and ability to perform harmful or unintended actions.

To ensure bacteria do not survive after excretion, a temperature-sensitive kill-switch ("cryodeath") was chosen to meet the safety requirement [18]. The kill-switch was designed to be stable (inactive) at temperature ranges normally found inside cattle digestive systems (around 37°C). It begins activating at temperatures below this. For our purposes, we use room temperature of 22°C and below as the kill-switch trigger.

Figure 1 shows an example of how these kinds of kill-switches operate. On the left side of this figure (normal rumen temperature range) there is an equilibrium between kill-switch toxin and antitoxin production; *i.e.*, the engineered cells continue to live inside the intestine, as intended. However, when the temperature drops below 22°C (right side of figure), the production of the toxin increases, causing the cells to intentionally die.

Clearly, prior to deploying mechanisms like toxin/antitoxin kill-switches, their designers should be able to provide evidence to certify that they are safe and secure for use in their intended environments. However, there is currently a lack of guidance as to what type of artifacts are appropriate for attesting to the safety and security of synthetic biology applications, and how this information can be identified, structured, and communicated. It is this gap which we seek to address with SynBioTrace.

III. SynBioTrace

We now present SynBioTrace, a four-step process that uses the information from fault trees and attack-defense trees to develop assurance cases addressing both safety and security. Figure 2 shows an overview of the process, which begins with the generation of a fault tree and attack-defense tree.

A. Step 1: Hazard Analysis

Hazard analysis is the foundation of building safe systems, so SynBioTrace begins there, as shown at the left of Figure 2. An assurance case for a synthetic biology application must consider failures that threaten its safety and attacks that threaten its security. The fault-tree analysis focuses on the internal logic (or computation) of the synthetic biology application's behavior. It considers logical faults that might be caused by missing/corrupt input or by unwanted events, and that contribute to hazardous failures. The attack-defense tree analysis addresses faults that can be caused by contextual elements and adjacent systems, including the physical environment, human agents, and manufacturing systems.

We have observed that fault-tree analysis is a form of hazard analysis that appears to be intuitive to synthetic biologists and feasible for them to build [13]. We speculate that this is because of its similarity to the top-down diagnostic visualizations using AND/OR trees often seen in laboratory notebooks.

As Knight has pointed out, "Attack trees provide a lot of insight into how an attacker might succeed just as fault trees provide a lot of insight into how a system might fail" [16]. Malicious attacks can cause critical computational failures. Attacks also can invalidate assumptions regarding run-time

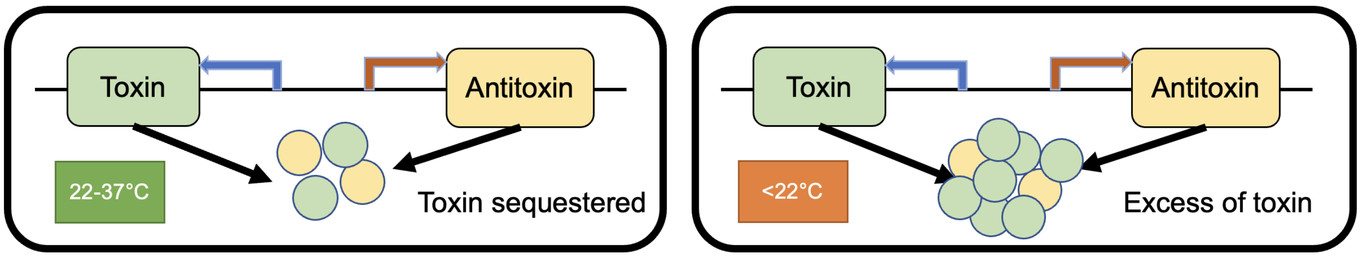

Fig. 1. The kill-switch produces roughly the same amount of toxins and antitoxins at the permissive temperature range (inside the digestive system), but an excess amount of toxins at and below (when excreted externally), killing the engineered bacteria. Image adapted from [18].

input, hardware platforms, and operational environments, creating risks to its safety. To investigate threats to the security of the cryodeath kill-switch, SynBioTrace incorporates an attack-defense tree analysis.

The outputs from Step 1 of Figure 2 are three development artifacts: a fault tree, an attack-defense tree, and an initial assurance case. These artifacts would be developed by synthetic biologists, potentially with help from safety and security experts. Their purposes are to: (1) specify requirements for the system to be developed, along with assumptions about their operational contexts; (2) identify feasible risks to system safety along with safety requirements to mitigate them; and (3) identify security attacks and mitigations to defend against them. Synthetic biology projects typically have much of this information in laboratory notebooks, but safety and security experts might need to help create appropriate documentation for assurance cases. The information gathered in Step 1 is used in Step 2.

B. Step 2: Goals and Context

In Step 2, shown at the mid-left in Figure 2, synthetic biologists, having generated the design of the DNA parts appropriate for the desired behavior of the engineered application, must connect the design verification with the information from the hazard analyses in Step 1. They use SynBioTrace to develop two intermediate trace tables showing the relationships between the artifacts. These tables add structure and rigor to the process of tracing hazards to mitigations and their implementation and validation. Table I shows the elements used to establish these traceability links in the fault-mitigation table and attack-defense table.

Column headings for these trace tables, shown in Table I, track relationships between the elements that derive from the fault-tree analysis and the attack-defense tree analysis:

- the events potentially causing the hazard
- the derived safety requirements to mitigate the hazard, together with the reasoning supporting why each requirement is justifiably believed to mitigate the hazard
- assumptions addressing leaf nodes related to the intended operational use of the system
- the cognizant agent responsible for confirming the traces
- the planned verification and validation to show that each requirement is satisfied in the implemented system
- the labeled nodes in the initial assurance case to which this element traces (empty until Step 3).

TABLE I
DESCRIPTION OF TRACE TABLES FOR SYNBIOTRACE.

Table Name	Lead Roles	Column Headings	Purpose
Fault-mitigation table	Synthetic Biologist or Safety Engineer	• Fault Nodes • Safety Requirements • Assumptions • Responsible Agents • Plan to Verify & Validate • Assurance Case Nodes	Plan strategies; identify contexts and assumptions; trace **mitigation** requirements to assurance case nodes
Attack-defense table	Synthetic Biologist or Security Engineer	• Attack Nodes • Defense Requirements • Assumptions • Responsible Agents • Plan to Verify & Validate • Assurance Case Nodes	Plan strategies; identify contexts and assumptions; trace **defense** requirements to assurance case nodes

The columns in the attack-defense tree are similar, as shown in the lower half of Table I and are not further explained here.

The output from Step 2 of Figure 2 is the two SynBioTrace tables: the fault-mitigation table and the attack-defense table. Populating these structured entries assists synthetic biologists in linking their hazard-based requirements to the verification artifacts they produce (*e.g.*, model simulations or laboratory experiments) for their application's design and implementation. The information gathered in Step 2 of SynBioTrace is then used in Step 3, with the verification and validation activities tracing to the assurance case's evidence nodes.

C. Step 3: Initial Assurance Case

Step 3, shown on the right of Figure 2 helps the synthetic biologist develop an initial assurance case. This is achieved by propagating information forward, gathered from the fault-mitigation table and the attack-defense table. Each leaf node identified in the tables should be traced to one or more elements in the assurance case as it is built up. The mitigation requirements traced from the fault tree support a sub-goal of safety, and the defense requirements traced from the attack-defense tree support a sub-goal of security. An advantage is that the initial assurance case can be started before evidence

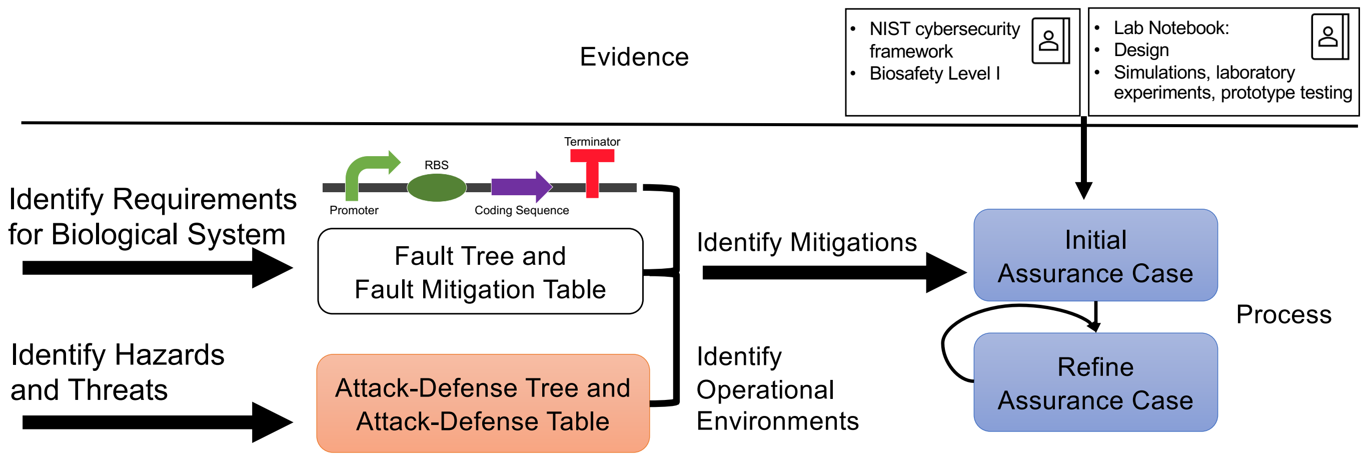
Evidence
NIST cybersecurity framework
Biosafety Level I
Lab Notebook:
Design
Simulations, laboratory experiments, prototype testing
Terminator
RBS
Promoter
Coding Sequence
Identify Requirements for Biological System
Fault Tree and Fault Mitigation Table
Identify Hazards and Threats
Attack-Defense Tree and Attack-Defense Table
Identify Mitigations
Identify Operational Environments
Initial Assurance Case
Refine Assurance Case
Process

Fig. 2. Four steps of the SynBioTrace process integrating the fault tree and attack-defense tree with biological safety requirements to show evidence of implemented mitigations in the assurance case. Mitigations are implemented in the design and code. An initial assurance case checks whether there is sufficient evidence that the product satisfies the safety and security requirements for the identified operational environments. The assurance case will likely need iterative refactoring because the application and environment typically evolve over time.

is available, and could be a tool for acquiring feedback from regulators or other stakeholders.

Every mitigation in the tables should lead to well-developed evidence elements within the assurance case. Rationales in the tables can suggest appropriate strategies for the assurance case. Assumptions about the operational environment, originating in the fault nodes and captured in the tables, are now considered for inclusion in the initial assurance case, depending upon the operational environment.

The tables from Step 2 thus serve as a useful mechanism for confirming that mitigations have been properly considered and their implementations verified with traces linking them to evidence elements in the assurance case. Note that although Step 3 considers fault-tree and attack-defense tree nodes separately, Step 4 will consider them jointly in order to further integrate the analysis of their relationships, and to consider potential trade-offs related to safety and security [14].

The output from Step 3 of Figure 2 is an initial assurance case for the synthetic biology application. The information and traceability relationships captured in Step 2 assist synthetic biologists to systematically build up the initial assurance case. To provide backward traceability, the elements added to the initial assurance case are given unique labels that are now entered into the last column of the SynBioTrace tables created in Step 2. The initial assurance case is then used in Step 4.

D. Step 4: Refinement

Step 4 is an incremental process for refining and refactoring the assurance case by determining whether some evidence nodes can be combined, expanded, or marked as undeveloped. The nodes are also refined based upon updated experimental data or modified requirements.

For example, if the host organisms evolve or mutate to bypass a safety feature, it could mean that a mitigation is defeated or that a new fault exists. Explicitly recognizing

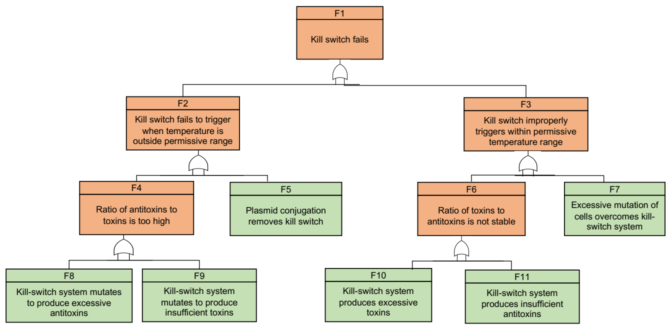
F1
Kill switch fails
F2
Kill switch fails to trigger when temperature is outside permissive range
F3
Kill switch improperly triggers within permissive temperature range
F4
Ratio of antitoxins to toxins is too high
F5
Plasmid conjugation removes kill switch
F6
Ratio of toxins to antitoxins is not stable
F7
Excessive mutation of cells overcomes kill-switch system
F8
Kill-switch system mutates to produce excessive antitoxins
F9
Kill-switch system mutates to produce insufficient toxins
F10
Kill-switch system produces excessive toxins
F11
Kill-switch system produces insufficient antitoxins

Fig. 3. Fault tree for the cryodeath kill-switch.

this could trigger either manual or automated processes to update system requirements which follow the traceability relationships, as well as facilitate cooperation across domains of expertise and provide greater modularity and reuse for engineers. Importantly, this also can provide regulators with improved coherence among the different artifacts needed to certify a cross-disciplinary, safety-critical system.

IV. Application of SynBioTrace

This section describes our application and evaluation of SynBioTrace on the kill-switch application. We provide additional artifacts and full-size figures on our supplementary website (http://sites.google.com/view/biotrace). We note that a complete analysis of the system in which the kill-switch operates would also include the methane-reduction system and supply-chain attacks.

A. Step 1: Hazard Analysis of Kill Switch

In accordance with the inputs to SynBioTrace shown in Figure 2, we identified the kill-switch's high-level safety requirement to be that it shall prevent the engineered bacteria from surviving after excretion, and its planned operational

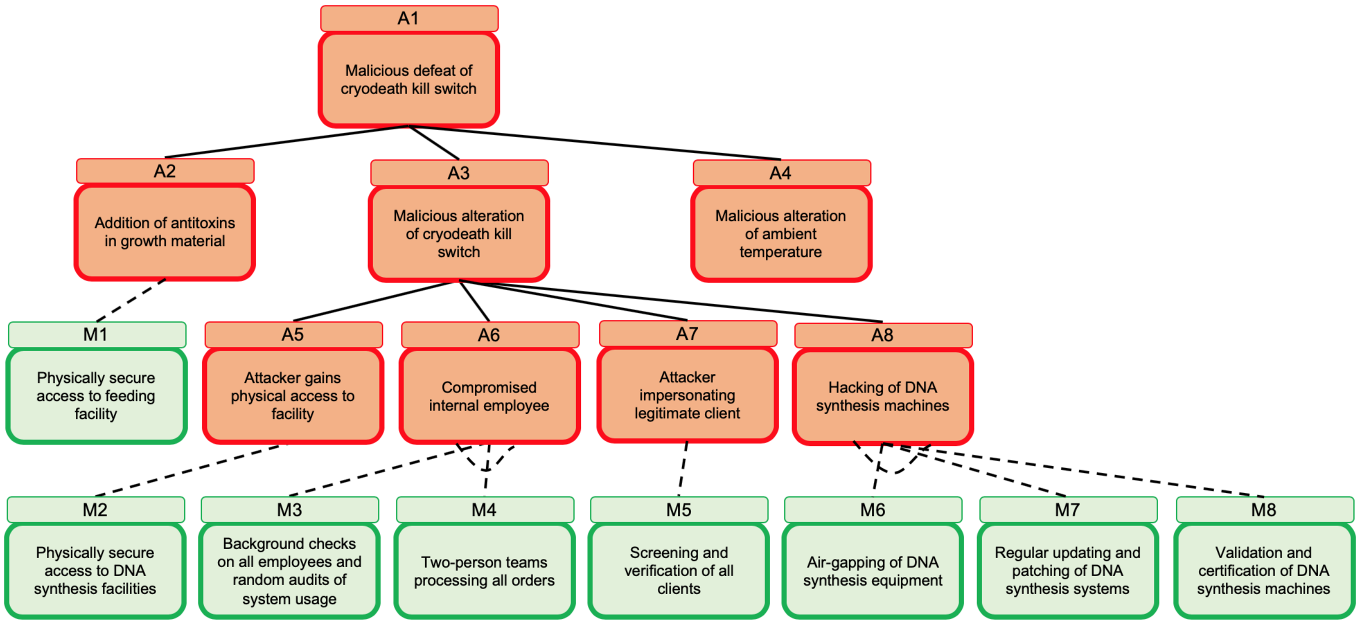
A1
Malicious defeat of cryodeath kill switch
A2
Addition of antitoxins in growth material
A3
Malicious alteration of cryodeath kill switch
A4
Malicious alteration of ambient temperature
M1
Physically secure access to feeding facility
A5
Attacker gains physical access to facility
A6
Compromised internal employee
A7
Attacker impersonating legitimate client
A8
Hacking of DNA synthesis machines
M2
Physically secure access to DNA synthesis facilities
M3
Background checks on all employees and random audits of system usage
M4
Two-person teams processing all orders
M5
Screening and verification of all clients
M6
Air-gapping of DNA synthesis equipment
M7
Regular updating and patching of DNA synthesis systems
M8
Validation and certification of DNA synthesis machines

Fig. 4. Attack-defense tree for the cryodeath kill-switch. Red nodes with an "A" label represent attack vectors and green nodes with an "M" label represent mitigations of those attack vectors.

context to be that the organism is *E. coli* strain Dh10β and the operating temperature is 22-37°C. The hazard of concern is that a failure of the kill-switch could cause engineered bacteria to survive after excretion, violating the system-level safety requirement described above.

As in Figure 2, we performed a fault-tree analysis (Figure 3), finding feasible events that could cause a kill-switch failure. These involve either the DNA of the kill-switch mutating, or the cells themselves mutating to bypass the kill-switch by not passing the kill-switch's code to the cells' children.

We developed the attack-defense tree (Figure 4) to identify malicious attacks that could cause the kill-switch to trigger when it should not, and attacks that could prevent the kill-switch from triggering when it should. This analysis found another five additional ways that attacks could cause failure of the kill-switch. Figure 4 shows that certain malicious actions violate assumptions about the operational environment, such as that a malevolent agent could disable the kill-switch by hacking the machine that synthesized (manufactured) it or by gaining access to the cattle's feeding facility.

We identified eight mitigations, M1-M8, that defended against these attacks. These involved securing access to the physical facilities (manufacturing and feeding), screening users and operators, qualifying and certifying hardware, and maintaining the manufacturing software. (Node A4 was judged to be too improbable to justify mitigation resources.)

B. Step 2: Goals and Context of Kill-Switch

In Step 2, we documented the relationships between the elements that derived from the fault-tree analysis in a fault-mitigation table. This included mitigations, rationales, and assumptions, *etc*. as described in Table I. We also developed an attack-defense table based on the attacks and mitigations from the attack-defense tree. We anticipate that populating these tables would involve discussions with other domain experts (*e.g.*, other synthetic biologists) to determine whether the mitigation requirements are adequate and feasible, and if they are not, appropriate updates to the tables would be necessary. The agents identified as responsible for the planned mitigations in the table may be software or hardware processes, regulatory standards, or individuals. These will ultimately provide evidence to support whether assurance claims have been met.

C. Step 3: Initial Assurance Case for Kill-Switch

In this SynBioTrace step, we generated an initial, candidate assurance case. It incorporated sub-goals for safety and security, and evidence node(s) for every leaf node in the tables. Evidence came from domain-specific knowledge and from documented experimental results.

We do not show the initial safety case here, but instead provide our refined candidate assurance case in Step 4 below. In Step 3 we also labeled each node in the initial assurance case and updated the information in the fault-mitigation and attack-defense tables with the appropriate traceability links.

D. Step 4: Refinement of Kill-Switch Assurance Case

In Step 4 of SynBioTrace, we, playing the role of a synthetic biologist, iterated over evidence nodes to identify which could be combined or refactored. First, we collapsed some evidence nodes with guidance provided by the trace tables where repetition of mitigation requirements or responsible agents suggested which nodes were good candidates for combination. Some nodes trace to multiple evidence nodes, while other nodes may be deleted or marked as undeveloped. We found

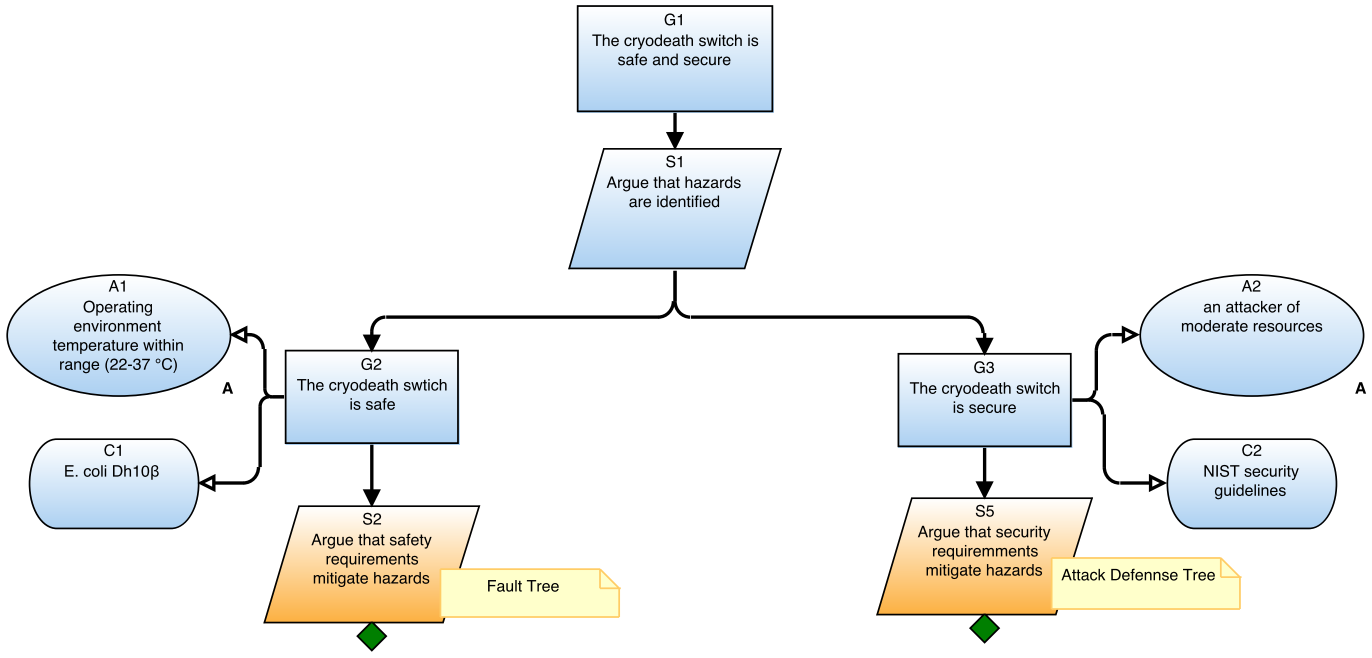
G1
The cryodeath switch is safe and secure
S1
Argue that hazards are identified
A1
Operating environment temperature within range (22-37 °C)
A
G2
The cryodeath swtich is safe
C1
E. coli Dh10β
S2
Argue that safety requirements mitigate hazards
Fault Tree
G3
The cryodeath switch is secure
A2
an attacker of moderate resources
A
C2
NIST security guidelines
S5
Argue that security requirements mitigate hazards
Attack Defennse Tree

Fig. 5. A candidate assurance case derived from the fault-mitigation table and attack-defense table, then refined to eliminate unwanted redundancies. The orange nodes are expanded in detail in Figures [6] and [7].

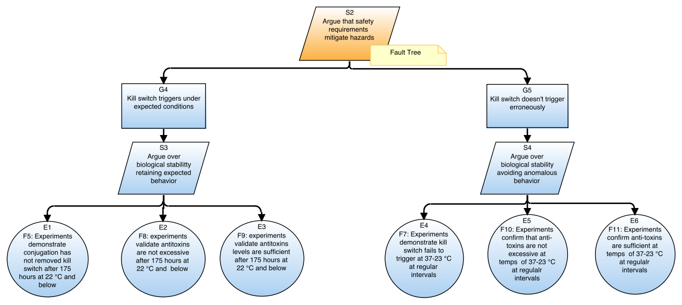
S2
Argue that safety requirements mitigate hazards
Fault Tree
G4
Kill switch triggers under expected conditions
G5
Kill switch doesn't trigger erroneously
S3
Argue over biological stabilitty retaining expected behavior
S4
Argue over biological stability avoiding anomalous behavior
E1
F5: Experiments demonstrate conjugation has not removed kill switch after 175 hours at 22 °C and below
E2
F8: experiments validate antitoxins are not excessive after 175 hours at 22 °C and below
E3
F9: experiments validate antitoxins levels are sufficient after 175 hours at 22 °C and below
E4
F7: Experiments demonstrate kill switch fails to trigger at 37-23 °C at regular intervals
E5
F10: Experiments confirm that anti-toxins are not excessive at temps of 37-23 °C at regulalr intervals
E6
F11: Experiments confirm anti-toxins are sufficient at temps of 37-23 °C at regualr intervals

Fig. 6. This is the safety piece of the assurance case continued from the top level. The orange nodes are from Figure [5].

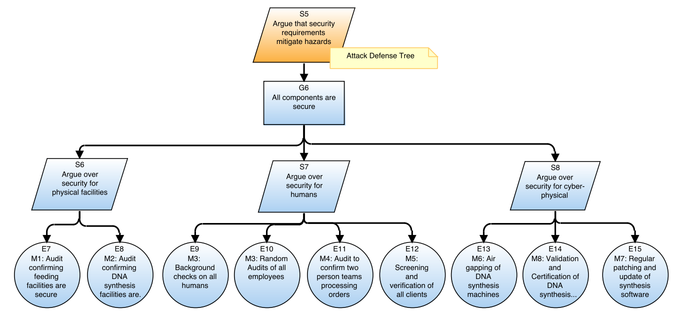
S5
Argue that security requirements mitigate hazards
Attack Defense Tree
G6
All components are secure
S6
Argue over security for physical facilities
S7
Argue over security for humans
S8
Argue over security for cyber-physical
E7
M1: Audit confirming feeding facilities are secure
E8
M2: Audit confirming DNA synthesis facilities are.
E9
M3: Background checks on all humans
E10
M3: Random Audits of all employees
E11
M4: Audit to confirm two person teams processing orders
E12
M5: Screening and verification of all clients
E13
M6: Air gapping of DNA synthesis machines
E14
M8: Validation and Certification of DNA synthesis...
E15
M7: Regular patching and update of synthesis software

Fig. 7. This is the security piece of the assurance case continued from the top level. The orange nodes are from Figure [5].

no safety or security conflicts in the kill-switch; however, guidance exists to handle them in assurance cases if they are discovered [14], [19], and their resolution could lead to deletion or modification of nodes. Finally, we updated the assurance-node columns in Table [I] to preserve round-trip traceability among the artifacts and their elements.

Figures [5], [6], and [7] show our candidate assurance case produced using SynBioTrace. The assurance case was built based on the final Step 4 refactoring, using the AdvoCATE tool from Denney and Pai [9].

Our intent is that a regulator should be able to review the SynBioTrace artifacts (the fault-tree and the attack-defense tree, their associated trace tables, and the refined version of the initial assurance case built from them) to readily check that all faults have been properly and adequately mitigated for the synthetic biology application.

V. Discussion

Observations from our application of SynBioTrace to the synthetic biology kill-switch case study include:

1) Fault trees and attack-defense trees each provided their own unique benefits for developing an assurance case. The fault-tree analysis considered events that could cause the switch's logic to fail directly, *e.g.*, mutation of either the switch or the system in which it is embedded, while the attack-defense tree considered malicious events that could break assumptions about the manufacturing or operational environment, *e.g.*, disabling the kill-switch by posing as an imposter operator.
2) Even the relatively simple case study here needed more domain expertise than anticipated. This inclines us to propose that synthetic biologists take lead in developing the initial assurance case, assisted by safety experts.
3) Our experience confirmed advice in the literature to start early and to build the assurance case incrementally [16],

[20], [21]. The best case would be where, as suggested above, a synthetic biologist has primary responsibility for producing the assurance case, so development of the application and the assurance case can be performed somewhat in tandem. SynBioTrace is designed to assist with this likely scenario.

4) Using tree structures for the safety and security analyses and for the initial assurance case facilitated readability and two-way traceability of the links among the artifacts, as well as providing support for future automation.
5) Our experience confirmed reports by others that it is hard to find the right balance between too much and too little information in creating an assurance case. We chose to keep the argument structures simple for review and maintenance purposes. However, more explanation was occasionally needed to establish why a certain evidence node sufficed to satisfy a sub-goal. Specifically for synthetic biology, we see a need to incorporate probability measures into the fault-tree analysis and assurance case. We agree that assurance cases can provide a unified view that helps manage uncertainty [22].

Two limitations of this study are that we, as researchers with some background in synthetic biology and molecular programming, simulated roles as synthetic biologists. The SynBioTrace process should be tested in a more realistic setting to determine its usefulness and effectiveness with full-time practitioners of synthetic biology. Although SynBioTrace was only applied to the kill-switch, it is fairly representative of other synthetic biology systems with safety and security aspects that we are likely to see deployed in the near future.

VI. Related Work

In related work, the AMASS tool platform adopts and extends existing technologies for management and assurance of compliance for cyber-physical systems [23]. It can integrate artifacts from other platforms to provide traceability for meeting standards and regulations, and supports evidence management and assurance case specification. It is intended for traditional cyber-physical systems and platforms rather than synthetic biology. Others have proposed more formal methods for integrating attack trees with fault trees [24].

Johnson and Kelly identified the challenges of co-assurance across the domains of safety and security and introduced the Safety-Security Assurance Framework (SSAF) to better integrate them [14]. Extension of SSAF to synthetic biology applications would be an interesting research direction.

There has been some prior work on applying assurance cases to synthetic biology. Cohen, *et al.*, proposed the use of a dynamic assurance case, called an assurance timeline, to address biological evolution [12]. In follow-on work, Firestone and Cohen suggested using parameterized patterns called assurance recipes to help non-experts build assurance cases [11]. We anticipate that ongoing work on formal verification of molecular devices [4], [25], [26] will lead to use of such evidence in synthetic biology assurance cases. Lutz proposed the use of a molecular program's artifacts, including its safety requirements and model verification results, as building blocks for preliminary safety-case arguments and evidence [13].

There are many extensions for assurance cases aimed at facilitating their creation and improving their overall quality and usage. Denney and Pai created a foundation for producing hierarchical cases through formal patterns [10]. Their AdvoCATE tool now supports bowtie diagrams and analysis, including trace links to hazard tables [27]. A survey by Makismov, Kokaly, and Chechik describes the various tools available which support assurance case assessment [28]. Chowdhury, *et al.*, provided criteria for developers and external reviewers as guidance for evaluating the structure and content of assurance cases [29].

The field of "cyberbiosecurity" has been rapidly growing, with some suggesting the discipline of synthetic biology developed with a naïve sense of trust [7], [30]. One research team has demonstrated the potential for a malicious actor to compromise DNA synthesis equipment by sending DNA samples tailored to compromise commonly used software [31].

Recent work to design a language called CRN++ provides a comprehensive framework for molecular programming that enables formal verification [32]. Domain-specific languages which allow for formal verification through languages like CRN++ will be an important component for certifying dependability by providing evidence for the assurance case nodes.

Similar work has also been proposed for helping stakeholders from different disciplines create assurance cases for AI-enabled systems. Kass *et al.*, suggested a "sociotechnical" approach to help satisfy regulatory expectations by creating assurance cases to support justifying the fairness of such systems throughout their lifecycles [33].

VII. Conclusions and Future Work

In this work we presented SynBioTrace, a four-step process to build assurance cases for synthetic biology applications that incorporates both safety and security requirements. Aimed at use by synthetic biologists, SynBioTrace starts from fault trees and attack-defense trees and uses them to guide development of an initial assurance case which can then be refactored. These artifacts are integrated and traced to the resulting assurance case, providing concrete evidence for mitigations. We performed a case study on a synthetic biology kill-switch to demonstrate SynBioTrace's feasibility. The evaluation results suggest the potential for SynBioTrace to improve the safety and security of future synthetic biology applications and to make it easier to build their assurance cases.

Acknowledgment

This work was supported in part by National Science Foundation Grants CCF-1901543, CCF-1909688, CCF-1900716, CCF-1513717, and by the National Institute of Justice Grant 2016-R2-CX-0023. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or National Institute of Justice. We thank Ewen Denney for providing the AdvoCATE tool.

References

[1] D. E. Cameron, C. J. Bashor, and J. J. Collins, "A brief history of synthetic biology," *Nature Reviews Microbiology*, vol. 12, no. 5, p. 381, 2014.

[2] W. Weber and M. Fussenegger, "Emerging biomedical applications of synthetic biology," *Nature Reviews Genetics*, vol. 13, no. 1, p. 21, 2012.

[3] G. Baldwin, *Synthetic Biology: A Primer*. World Scientific, 2016.

[4] S. J. Ellis, T. H. Klinge, J. I. Lathrop, J. H. Lutz, R. R. Lutz, A. S. Miner, and H. D. Potter, "Runtime fault detection in programmed molecular systems," *ACM Transactions on Software Engineering Methodology*, vol. 28, no. 2, pp. 6:1–6:20, 2019.

[5] W. E. Forum, "Biosecurity innovation and risk reduction: A global framework for accessible, safe and secure DNA synthesis," 2020.

[6] G. N. Mandel and G. E. Marchant, "The living regulatory challenges of synthetic biology," *Iowa L. Rev.*, vol. 100, p. 155, 2014.

[7] J. Firestone, "The need for soft law to regulate synthetic biology," *Jurimetrics*, vol. 60, pp. 139–173, 2020.

[8] D. J. Rinehart, J. C. Knight, and J. Rowanhill, "Current practices in constructing and evaluating assurance cases with applications to aviation," NASA, Tech. Rep., 2015.

[9] E. Denney and G. Pai, "Tool support for assurance case development," *Autom. Softw. Eng.*, vol. 25, no. 3, pp. 435–499, 2018.

[10] E. W. Denney and G. J. Pai, "Safety case patterns: theory and applications," 2015.

[11] J. Firestone and M. B. Cohen, "The assurance recipe: Facilitating assurance patterns," in *Computer Safety, Reliability, and Security - SAFECOMP*. Springer International Publishing, 2018, pp. 22–30.

[12] M. B. Cohen, J. Firestone, and M. Pierobon, "The assurance timeline: Building assurance cases for synthetic biology," in *Computer Safety, Reliability, and Security - SAFECOMP*. Springer International Publishing, 2016, pp. 75–86.

[13] R. R. Lutz, "Requirements engineering for safety-critical molecular programs," in *30th IEEE International Requirements Engineering Conference, RE 2022, Melbourne, Australia, August 15-19, 2022*. IEEE, 2022, pp. 302–308.

[14] N. Johnson and T. Kelly, "Devil's in the detail: Through-life safety and security co-assurance using SSAF," in *Computer Safety, Reliability, and Security*. Springer International Publishing, 2019, pp. 299–314.

[15] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, 2012.

[16] J. Knight, *Fundamentals of Dependable Computing for Software Engineers*. CRC Press, 2012.

[17] "UNL 2017 iGEM Team Page," http://2017.igem.org/Team:UNebraska-Lincoln, last accessed: January 26, 2020.

[18] F. Stirling, L. Bitzan, S. O'Keefe, E. Redfield, J. W. Oliver, J. Way, and P. A. Silver, "Rational design of evolutionarily stable microbial kill switches," *Molecular Cell*, vol. 68, no. 4, pp. 686 – 697, 2017.

[19] I. Friedberg, K. McLaughlin, P. Smith, D. M. Laverty, and S. Sezer, "STPA-SafeSec: Safety and security analysis for cyber-physical systems," *Journal of Information Security and Applications*, vol. 34, pp. 183–196, 2017.

[20] T. Kelly and R. Weaver, "The Goal Structuring Notation – A Safety Argument Notation," in *Proceedings on Dependable Systems and Networks*, 2004.

[21] R. R. Lutz and A. Patterson-Hine, "Using fault modeling in safety cases," in *19th Int'l Symposium on Software Reliability Engineering, ISSRE*, 2008, pp. 271–276.

[22] M. Chechik, R. Salay, T. Viger, S. Kokaly, and M. Rahimi, "Software assurance in an uncertain world," in *Fundamental Approaches to Software Engineering - 22nd Int'l Conf FASE, ETAPS*, ser. LNCS, vol. 11424, 2019, pp. 3–21.

[23] J. L. de la Vara, A. Ruiz, and G. Blondelle, "Assurance and certification of cyber-physical systems: The AMASS open source ecosystem," *J. Syst. Softw.*, vol. 171, p. 110812, 2021.

[24] I. Nai Fovino, M. Masera, and A. De Cian, "Integrating cyber attacks within fault trees," *Reliability Engineering & System Safety*, vol. 94, no. 9, pp. 1394–1402, 2009, eSREL 2007, 18th European Safety and Reliability Conference.

[25] L. Cardelli, M. Kwiatkowska, and M. Whitby, "Chemical reaction network designs for asynchronous logic circuits," *Natural Computing*, vol. 17, no. 1, pp. 109–130, 2018.

[26] J. I. Lathrop, J. H. Lutz, R. R. Lutz, H. D. Potter, and M. R. Riley, "Population-induced phase transitions and the verification of chemical reaction networks," *Natural Computing*, vol. 23, no. 2, pp. 347–363, 2024.

[27] E. Denney, G. Pai, and I. Whiteside, "The role of safety architectures in aviation safety cases," *Reliability Engineering & System Safety*, vol. 191, p. 106502, 2019.

[28] M. Maksimov, S. Kokaly, and M. Chechik, "A survey of tool-supported assurance case assessment techniques," *ACM Computing Surveys*, vol. 52, no. 5, pp. 101:1–101:34, 2019.

[29] T. Chowdhury, A. Wassyng, R. F. Paige, and M. Lawford, "Criteria to systematically evaluate (safety) assurance cases," in *30th IEEE Int'l Symp on Software Reliability Engineering ISSRE*, K. Wolter, I. Schieferdecker, B. Gallina, M. Cukier, R. Natella, N. Ivaki, and N. Laranjeiro, Eds. IEEE, 2019, pp. 380–390.

[30] J. Peccoud, J. E. Gallegos, R. Murch, W. G. Buchholz, and S. Raman, "Cyberbiosecurity: from naive trust to risk awareness," *Trends in biotechnology*, vol. 36, no. 1, pp. 4–7, 2018.

[31] P. Ney, K. Koscher, L. Organick, L. Ceze, and T. Kohno, "Computer security, privacy, and DNA sequencing: Compromising computers with synthesized DNA, privacy leaks, and more," in *26th USENIX Security Symposium*, 2017, pp. 765–779.

[32] M. Vasic, D. Soloveichik, and S. Khurshid, "CRN++: molecular programming language," *Nat. Comput.*, vol. 19, no. 2, pp. 391–407, 2020.

[33] M. H. Kaas, C. Burr, Z. Porter, B. Ozturk, P. Ryan, M. Katell, N. Polo, K. Westerling, and I. Habli, "Fair by design: A sociotechnical approach to justifying the fairness of ai-enabled systems across the lifecycle," *arXiv preprint arXiv:2406.09029*, 2024.