MP 180674
MITRE PRODUCT

MITRE

TRACE – Traversal-driven Risk Assessment of Composite Effects

Project No.: 10AOH810-JF

Lexington Park, MD

Author:

F. Lynam

Peer Review:

B. Race
L. Cooper
L. Oringer
R. McInnes
M. McFarren
A. McEachron

October 2019

Abstract

This paper describes TRACE, a method of constructing a mathematical model that describes a mixture of stochastic and deterministic steps that can be taken in a process. Similar to a Markov Chain and similar graph theoretical models, TRACE is designed specifically to accommodate asymmetric threat vector processes, such as cyber offensive or adversarial supply chain operations. Unlike other methods, TRACE does not constrain process execution to a stateful representation, meaning that it can account for adversarial activities that may require execution of many simultaneous paths. This paper argues that the ability to address non-stateful effects is functionally significant to this problem space. This model construct is meant to be used to analyze both for weaknesses in the design of socio-technical systems and to assess for the effectiveness of corrective or mitigating courses of actions. This paper is targeted at a security analyst who wants to understand the considerations that apply to how TRACE models work.

Overview

TRACE is a stochastic adversarial process analysis. It supports performing adversarial decision making analysis based on how likely the available options are to succeed instead of relying on inferred policy driving agency. It allows for analysis of highly complex processes by understanding the simple rules governing each choice along each path. TRACE provides a mathematical framework to optimize decision making to maximize the odds of successfully executing a desirable process, like a mission, or preventing an undesirable one, like a cyber weapon. Using a TRACE model allows for use of the existing TRACE tools for trade space analysis.

Probabilistic Treatment of Processes

TRACE can be described mathematically as a probabilistic treatment for non-stochastic processes, such as open-ended decision making. Consider a game with a quarter and a nickel where the player wins upon flipping either coin and obtaining a heads. What are the odds that the player will win?

If the player must flip one and only one coin, the odds are 50%. In statistics, this is represented by the statement that P[heads] = 50%, notionally read as “the probability of heads is 50%”. However, if the player may flip as many coins as they like, but only once, the odds range from 0% to 75% depending on the player’s choices. This is represented in statistics by three different Boolean algebra functions:

No coins: P[heads] = 0.

You cannot win if you do not play.

One coin: P[heads] = 50%.

By definition.

Both coins: P[heads|quarter] + P[heads|nickel] = 75%.

If the player can flip both coins, the odds for either being heads is 50%. Thus, the odds for neither being heads is 50% * 50% = 25%. Consequently, the probability of getting at least one heads given the player can flip both the quarter and the nickel is 75%.

These Boolean functions can be represented as graphs in a logic diagram using an “OR” gate:

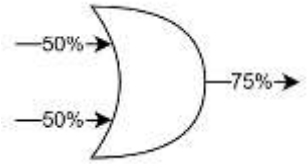

Figure 1

These coin flips can also be described as a stochastic process:

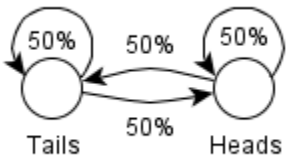

Figure 2

A coin exists as either a heads or a tails, denoted as "states" by the circle-shaped "nodes" in this graph. In graph theory, moving from node to node along these "edges" is called graph traversal. When a flip occurs, it can either traverse the graph to the other state or retain its current state, with equal probability. A stochastic process is a description of how a system changes states where state transitions are completely described by mutually exclusive probabilities: a flipped coin either becomes a heads or a tails. Coin flips are a type of stochastic process called Markov chains. A Markov chain is a special case of stochastic processes which are memoryless: it doesn't matter whether it was a heads the last ten times, the odds of getting heads again are the same.

TRACE Graphs

Unfortunately, decision making isn't a stochastic process. The odds of getting a heads should the player choose to flip the coin is a stochastic process, but the choice is not.[1] In TRACE, this decision process flow is therefore illustrated as a directed graph whose edges have probabilities that reflect the odds of successfully moving to the next desired state, should an attempt be made:

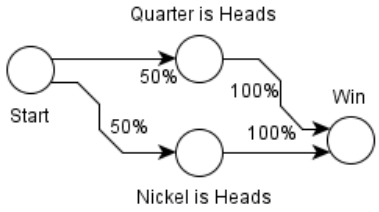

Figure 3

This graph is simply another representation of the same concepts captured in the other two forms (Figures 1 and 2), except that in contrast to the stochastic process, this graph is about completion toward the player's objective, and therefore only includes the winning scenarios. The graph describes all possible progression states and the order they occur for every possible variant of one attempt at the game. The player may make it to the winning condition. They may exhaust all options available at some state and stop there. Perhaps they get a chance to play again later.

The player starts at the "Start" node, with the choices of flipping either coin before them. Either or both can be selected, and success with either results in reaching the "Win" node. Describing the probability of winning requires exhaustively searching for every possible path and overlaying them across each other. This becomes non-trivial when more complicated graphs contain cycles, or multiple "starting" or "winning" conditions.

Solving the TRACE graph is first focused on describing the entire space of adversarial options, but particularly focused on bounding the worst case adversary choice. For example, the probability of winning in this game is dependent upon the player's choice of actions. Flipping both coins results in a 75% chance, either results in two options at 50% each, and flipping neither results in a 0% chance. Imagine being someone who wanted the player to lose: a defender. Without sufficient evidence about

[1] G. Wyss, J. Clem, et al, "A Method for Risk-Informed Management of Enterprise Security (RIMES)," Sandia National Laboratories, October 2010

which option the player will select, all four outcomes are possible. The defender can only state with confidence that the odds the player will lose are at least 25%.

In the TRACE graph, the conservative solution for the probability of reaching the “Win” state is the product of the probabilities of all edges along any path to reach that state. This allows for a single graph to express all player choices. As the graph by design contains all possible winning paths, TRACE provides an overall residual risk independent of adversarial decision making.

This is similar to graphs like attack trees and decision trees, which can also be used to describe non-stochastic processes whose state transitions can sometimes be neither mutually exclusive nor memoryless. These factors allow for the generation of models which represent a real process flow, vice representing the mathematics behind it. Applying the TRACE analysis to this graph produces the same mathematical result as a stochastic treatment of the individual events or a logic diagram (such as Figure 1), and is simply a means to depict complex sequences of events that is convenient both for human understanding and mathematical modeling.

For example, consider adding a blue and a red die to the game. Now, a player must roll anything greater than 1 on either die, then a heads on either coin. The TRACE model for this process can be simply and exhaustively depicted in a manner mirroring the above description without much additional abstraction:

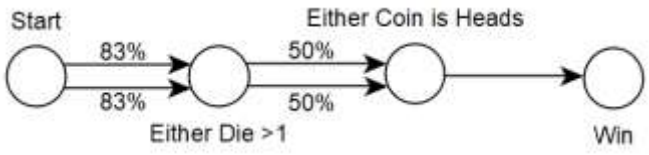

Figure 4

This game could also be modeled using a Markov chain, however this would be an inefficient model choice. Developing a Markov model for this process would require reinterpretation of the game rules into a state machine, modeling the outcome of the rules vice the rules themselves. As a rule set increases in scope, this rapidly becomes extremely difficult for a Markov chain formulation of this kind of problem. In particular, game playing is generally not memoryless, requiring a unique state in the Markov model for every possible combination of elements and some mathematically difficult method to construct the acceptable edges between those states as well as to calculate their respective transition probabilities.

While all of the information to construct a Markov chain model can be extracted from the TRACE graph, the relevant information to the question is already fully contained without that translation. The TRACE model formulation is better suited to examining the question of interest than a Markov model would be as it effectively takes advantage of graph symmetry to avoid having to expand that state space at all, requiring less human interpretation of the problem definition to construct a useable model which can be readily solved through Monte Carlo methods.

Consider a different rule set that flips the coins, then rolls the die. If the blue die is two, the player gets to flip again. If the red die is six, the player wins. The TRACE model would be as follows:

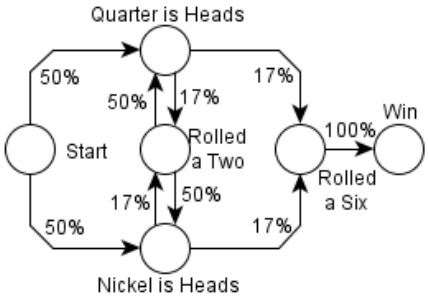

Figure 5

The allowed process paths are now cyclical. Each node and edge can still be described with simple rules, as done above, but now the interactions of those rules at a larger scale is much harder to understand. In this graph, there are essentially infinite possible paths, so long as the player gets lucky and keeps rolling a two. This is called a cycle, and while cycles make the computation of the overall probability of winning challenging, the difficulty of the math does not change the nature of the rules. Numerical approximation methods can be applied to solve this type of problem, even when pure algebraic solutions are challenging or may not even exist.

TRACE allows humans to focus on the details that matter (what the particular rules are) and puts the computationally intensive work (dealing with all the paths) squarely in the hands of the computer. This facilitates increased focus on defining the right rules that represent the process, without expending human intellect on dealing with the explosive scale of all the combinations of possibilities and states. Further, in conjunction with analytics that show dominant nodes or paths, TRACE guides research and analysis to explore less well-defined processes and concentrates on asking useful questions, such as assessing the probability of individual important steps, or validating that key anticipated states are actually attainable.

Application to Specific Processes

TRACE provides analytics about paths and nodes in a process graph for particular kinds of processes that apply well to asynchronous games like threat vector processes. However, the manner in which those paths are selected, traversed and constructed can vary based on the specific context of the analysis.

Decision Making Policy

The criteria used by a player to select a path through a game is often referred to as “policy.” Game theory approaches, which fundamentally examine particular traversal sets through a graph like a TRACE graph, tend to emphasize particular policy approaches. Often, the policy selected is that the player will invest in the shortest path, or have some finite resources and invest in a limited set of paths. These methods can be evaluated against TRACE graphs of games.

However, for asymmetric threat vector space analysis such as cyber attack decomposition against an adaptive adversary, defining defensive options in terms of these offensive approaches produce iteratively incomplete solutions. Removing the shortest path in a game may simply leave a path nearly

as short untouched. The most effective defensive solution when assessing how changes to a graph may reduce the probability of successful traversal is to treat the adversary as investing in all options available, and iterate on design options which effectively reduce the probability of successful traversal to some acceptable value given that assumption leveraging analytics that incorporate all paths at once are employed. TRACE functionally acts as an asynchronous game in this way, where the adversarial player is allowed to take as many turns as they might decide to invest in.

This is in contrast to more common "turn taking" approaches to game theory, which require iteration using some restricted adversary policy such as trimming the shortest path. Given a TRACE graph with millions of possible paths, it's not clear that such an approach could ever find an efficient solution to trim a large number of vectors. Consequently, by leveraging an unrestricted adversarial investment assumption, analysis is able to produce measures which promote structuring a graph to have no available options in fewer iterations.

Time-Varying Functions

The probability of traversing a particular edge may not be a static value. For example, consider this same game, except the player can try again once per day and retains persistence at every state ever attained. This means that once a position in the TRACE model is reached, that position is retained on the start of the next day. The probability of moving along any individual edge can be described as a time-varying function. The odds of getting heads on the quarter are 50% by the first day (one attempt at 50%), 75% by the second (two attempts at 50% each). The odds of winning the game on any given day increase over time, and vary with the choices the player makes on which options to attempt.

Notably, using persistence in the time-varying case eliminates the need to attempt the edges which return to previously obtained states: if the player flips a quarter and gets heads, and then rolls a two, this eliminates the need to attempt the path back to the quarter. This aligns well with an idea sometimes called the "monotonically increasing intrusion depth" assumption in a cyber weapon process, where it is deemed credible that an adversary will not need to develop two novel exploits to reacquire the same target as part of an attack.

Time-Sequence Dependence

As another special case, consider a game where all of the state transitions can be attempted simultaneously or out of sequence, and a winning scenario happens once any set of state transitions has occurred that completes a path to winning. Here, this distinction is defined as time-sequence dependence or independence.

The process in figure 4 was introduced as a time-sequence dependent process. This is important when determining the odds of completing a process when multiple attempts are available to a player who retains obtained states. If on day one the player flips a heads, then they get to roll a die on day one and day two. However, if the player flips a tails on day one, they don't get to roll a die on day one. The ability to roll a die is dependent on attaining the preceding state, of having flipped a heads.

This is different than if the player throws the dice and flips the coins at the same time, and only re-rolls or re-flips if they didn't get a six or a heads. In this case, the odds of reaching a set of conditions that completes a path through the model are different, because the acceptable process rules are different. This is a time-sequence independent process, and is valid for describing processes such as cyber weapon

development, which can cobble together individual pieces of an attack chain after they've been independently developed. TRACE models can be applied for either of these types of processes, and the nature of the process should be considered as part of model construction.

Conclusion

TRACE graphs are a particular formulation of process graphs comparable to Markov decision processes but tailored to the specific needs of asynchronous game playing problems like asymmetric threat vector analysis. By design, analysis of these graphs should focus less on selecting the best path for attack and more on the set of nodes most likely to be involved in a successful attack. Analysis of this kind allows for more robust, failure-tolerant planning in both offensive and defensive activities.