



دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر



توضیح خطا در شبکه‌های مبتنی بر نرم‌افزار با استفاده از استدلال مبتنی بر علیت

پایان‌نامه برای دریافت درجه کارشناسی ارشد در رشته مهندسی کامپیوتر
گرایش نرم‌افزار

امیرحسین صیحانی

اساتید راهنما

دکتر حسین حجت و دکتر محمدرضا موسوی

شهریور ۱۴۰۱

چکیده

واژگان کلیدی

فهرست مطالب

۳	فصل ۱: تعاریف و دانش پیش زمینه
۳	۱.۱ مقدمه
۳	۲.۱ شبکه‌های مبتنی بر نرم افزار
۴	۳.۱ نتکت
۵	۱.۳.۱ دستور زبان نتکت
۵	۲.۳.۱ معنای نتکت
۷	۳.۳.۱ توصیف رفتار شبکه با نتکت
۱۰	۴.۳.۱ درستی سنجی برنامه‌های نتکت
۱۱	۴.۱ نتکت پویا
۱۱	۱.۴.۱ دستور زبان نتکت پویا
۱۲	۲.۴.۱ معنای عملیاتی نتکت پویا
۱۴	۳.۴.۱ توصیف برنامه‌ها در نتکت پویا
۱۶	۵.۱ ساختمان رویداد
۱۸	۶.۱ مدل علی
۲۱	۱.۶.۱ علت واقعی
۲۳	۲.۶.۱ پیدا کردن علت واقعی در مسائل
۲۴	۳.۶.۱ مدل تعمیم یافته
۲۵	فصل ۲: روش و راه حل پیشنهادی
۲۵	۱.۲ مقدمه

۲۵	۲.۲ معنای عبارات نت‌کت پویا در قالب ساختمان رویداد
۲۷	۱.۲.۲ ترکیب ساختمان رویدادهای برچسب‌دار
۲۹	۲.۲.۲ معنای عبارات نت‌کت پویا در قالب ساختمان رویداد برچسب‌دار
۳۰	۳.۲ مدل علی برای ساختمان رویداد
۳۳	۴.۲ پیدا کردن علت خطا در نت‌کت پویا

مراجع اول

سوم واژه‌نامه فارسی به انگلیسی

پنجم واژه‌نامه انگلیسی به فارسی

فهرست کارهای باقیمانده

فصل ۱

تعاریف و دانش پیش زمینه

۱.۱ مقدمه

در این فصل مفاهیم مورد نیاز و استفاده در این پروژه مورد بررسی قرار می گیرند. این فصل شامل ۵ بخش است. ابتدا مفاهیم کلی شبکه های مبتنی بر نرم افزار بررسی می شوند. سپس زبان های نت کت^۱ و نت کت پویا^۲ که زبان های مورد استفاده در این تحقیق هستند شرح داده می شوند. در ادامه تعاریف اولیه ساختمان رویداد^۳ که به عنوان معنای زبان مورد استفاده در این تحقیق استفاده می شود شرح داده می شود. در بخش آخر مدل علی^۴ که در این تحقیق برای پیدا کردن علت واقعی خطا مورد استفاده قرار می گیرد توصیف شده است.

۲.۱ شبکه های مبتنی بر نرم افزار

شبکه های کامپیوتری یکی از زیرساخت های حیاتی سیستم های کامپیوتری هستند. با وجود اینکه نیازمندی های این شبکه ها بسیار بیشتر و پیشرفته تر نسبت به گذشته شده است تکنیک ها و متدهای مدیریت و ساخت آن ها همچنان مانند گذشته است. این مساله مدیریت آن ها را در استفاده های امروزی پیچیده و مستعد خطا می کند.

¹NetKAT

²DyNetKAT

³Event Structure

⁴Causal Model

حتی شرکت‌های بزرگی مانند، Amazon Github یا GoDaddy مرتباً مشکلاتی در شبکه‌های خود پیدا می‌کنند [۳]. شبکه‌های مبتنی بر نرم‌افزار^۵ یک پارادایم جدید برای طراحی و پیاده‌سازی شبکه‌های کامپیوتری هستند که علاوه بر اینکه مدیریت و درستی‌سنجی آن‌ها را با روش‌های اصولی‌تر امکان‌پذیر می‌کنند، انعطاف بالایی هم دارند. به صورت خلاصه در یک شبکه‌ی مبتنی بر نرم‌افزار رفتارهای کنترلی (تغییر و به روز رسانی قوانین ارسال^۶) از عناصر شبکه مانند سویچ‌ها یا روترها جدا می‌شوند و توسط یک کنترل‌کننده‌ی مرکزی انجام می‌شوند. به عنوان مثال در زبان OpenFlow [۶] رفتار سویچ‌های شبکه توسط تعدادی قانون به شکل تطبیق و اجرا توصیف می‌شوند. قوانینی به این شکل ابتدا بررسی می‌کنند که بسته با قانون تطابق داشته باشد و اگر تطابق داشت عملیات توصیف شده در قانون اجرا می‌شود. با ساده شدن عناصر شبکه، تمامی به‌روزرسانی‌ها و تغییرهای لازم در شبکه توسط یک کنترل‌کننده مرکزی انجام می‌شود. متمرکز شدن رفتار کنترلی سیستم باعث می‌شود تا استدلال و درستی‌سنجی در مورد رفتار شبکه آسان‌تر شود.

۳.۱ نت‌کت

نت‌کت^۷، یک زبان برای توصیف شبکه‌های مبتنی بر نرم‌افزار است [۱]. این زبان با وجود دستور زبان^۸ ساده‌ای که دارد، بر اساس KAT [۵] بنا شده و به همین دلیل یک سیستم معادلاتی صحیح و کامل^۹ دارد. این سیستم معادلاتی کمک می‌کند تا با استفاده از روش‌های جبری و اثبات تساوی برنامه‌های توصیف شده در این زبان، در مورد آن‌ها استدلال کرد.

^۵Software Defined Network

^۶Forwarding Rule

^۷NetKAT

^۸Syntax

^۹Sound and Complete

۱.۳.۱ دستور زبان نت‌کت

در نت‌کت هر بسته ^{۱۰} به عنوان یک نگاشت از یک مجموعه از فیلدهای f_1, f_2, \dots, f_n به اعداد طبیعی با تعداد ارقام ثابت در نظر گرفته می‌شود. آی‌پی ^{۱۱} های مبدا و مقصد، نوع بسته، پورت ^{۱۲} های مبدا و مقصد مثال‌هایی از این فیلدها هستند. دستور زبان نت‌کت به صورت زیر تعریف می‌شود:

$$a, b ::= 1|0|f = n|a + b|a \cdot b|\neg a$$

$$p, q ::= a|f \leftarrow n|p + q|p \cdot q|p^*|dup$$

۲.۳.۱ معنای نت‌کت

برای اینکه امکان استدلال در مورد مسیرهای طی شده توسط یک بسته در شبکه وجود داشته باشد، از مفهومی به نام تاریخچه‌ی بسته ^{۱۳} استفاده می‌شود. هر تاریخچه‌ی بسته، یک دنباله از بسته‌ها است که بسته نخست دنباله، به عنوان بسته‌ی فعلی در نظر گرفته می‌شود. به صورت شهودی عبارت‌های ۱ و ۰ به ترتیب به معنای عبور دادن ^{۱۴} و رها کردن ^{۱۵} بدون شرط بسته هستند. عبارت $f = n$ در صورتی بسته را عبور می‌دهد که مقدار فیلد f آن برابر با n باشد. عبارت $f \leftarrow n$ مقدار n را به فیلد f بسته اختصاص می‌دهد. عبارت‌های dup باعث می‌شوند تا یک کپی از بسته‌ی فعلی ایجاد شود و به تاریخچه‌ی بسته‌ها اضافه شود. این عبارات در رفتار شبکه تأثیری ندارند اما امکان استدلال در مورد تمامی تغییرات ایجاد شده در حین جابه‌جایی بسته در شبکه را فراهم می‌سازند. به

¹⁰Packet

¹¹IP

¹²Port

¹³Packet History

¹⁴Forward

¹⁵Drop

صورت دقیق، معنای هر عبارت ^{۱۶} نتکت با استفاده از معادلات زیر تعریف می شود:

$$\llbracket p \rrbracket H \in \mathcal{P}(H)$$

$$\llbracket 1 \rrbracket h \triangleq \{h\}$$

$$\llbracket 0 \rrbracket h \triangleq \{\}$$

$$\llbracket f = n \rrbracket (pk :: h) \triangleq \begin{cases} \{pk :: h\} & \text{if } pk.f = n \\ \{\} & \text{otherwise} \end{cases}$$

$$\llbracket \neq a \rrbracket \triangleq \{h\} \setminus (\{a\}h)$$

$$\{f \leftarrow n\}(pk :: h) \triangleq \{pk[f := n] :: h\}$$

$$\llbracket p + q \rrbracket h \triangleq \llbracket p \rrbracket h \cup \llbracket q \rrbracket h$$

$$\llbracket p \cdot q \rrbracket h \triangleq (\llbracket p \rrbracket \bullet \llbracket q \rrbracket)h$$

$$\llbracket p^* \rrbracket h \triangleq \bigcup_{i \in \mathbb{N}} F^i h$$

$$F^0 h \triangleq \{h\}$$

$$F^{i+1} h \triangleq (\llbracket p \rrbracket \bullet F^i)h$$

$$(f \bullet g)x \triangleq \bigcup \{g(y) | y \in f(x)\}$$

$$\llbracket dup \rrbracket (pk :: h) \triangleq \{pk :: (pk :: h)\}$$

¹⁶Expression

نت‌کت علاوه بر اصول موضوعه‌ی ^{۱۷} KAT اصول موضوعه‌ی زیر را هم شامل می‌شود تا دستگاه معادلاتی صحیح و کامل ^{۱۸} داشته باشد:

$$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad (۱.۱)$$

$$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad (۲.۱)$$

$$dup \cdot f = n \equiv f = n \cdot dup \quad (۳.۱)$$

$$f \leftarrow n \cdot f = n \equiv f \leftarrow n \quad (۴.۱)$$

$$f = n \cdot f \leftarrow n \equiv f = n \quad (۵.۱)$$

$$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n' \quad (۶.۱)$$

$$f = n \cdot f = n' \equiv 0, \text{ if } n \neq n' \quad (۷.۱)$$

$$\sum_i f = i \equiv 1 \quad (۸.۱)$$

اصل‌های ۱.۱، ۲.۱، ۳.۱، خواص جابه‌جایی ^{۱۹} عملیات‌ها را بیان می‌کنند. اصل ۴.۱ بیان می‌کند که اختصاص مقدار n به یک فیلد و سپس فیلتر کردن بر روی این فیلد با همین مقدار معادل با عملیات اختصاص ^{۲۰} به تنهایی است. مشابه همین اصل برای یک فیلتر و سپس یک اختصاص هم در اصل ۵.۱ مشخص شده. اصل ۶.۱ بیان می‌کند که در دنباله‌ای از اختصاص مقادیر به یک فیلد مشخص، تنها آخرین اختصاص تاثیر دارد. در اصل ۷.۱ مشخص شده است که مقدار یک فیلد نمی‌تواند دو مقدار متفاوت داشته باشد. در نهایت اصل ۸.۱ بیان می‌کند که عملیات مجموع فیلترها به ازای هر مقدار ممکن برای یک فیلد مشخص برابر عنصر همانی ^{۲۱} است.

۳.۳.۱ توصیف رفتار شبکه با نت‌کت

در ادامه نحوه‌ی توصیف یک شبکه با استفاده از نت‌کت بیان می‌شود. در شکل ۱.۱ شبکه شامل دو سویچ

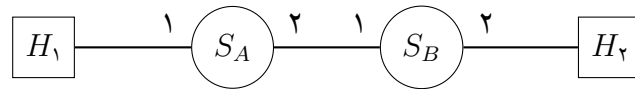
¹⁷ Axiom

¹⁸ Sound and Complete

¹⁹ Commutative

²⁰ Assignment

²¹ Identity



شکل ۱.۱: مثال شبکه

^{۲۲} A و B و دو هاست ^{۲۳} می باشد. هر سویچ دو پورت دارد که با شماره‌های ۱ و ۲ مشخص شده‌اند. در این شبکه هدف این است که امکان جابه‌جایی همه‌ی بسته‌ها به غیر از بسته‌هایی که از نوع SSH هستند وجود داشته باشد. عبارت نت‌کت زیر را در نظر بگیرید:

$$p \triangleq (dst = H_1 \cdot pt \leftarrow 1) + (dst = H_2 \cdot pt \leftarrow 2)$$

این عبارت همه‌ی بسته‌هایی که مقصد آن‌ها هاست ۱ باشد را به پورت ۱ و همه‌ی بسته‌هایی که مقصد آن‌ها هاست ۲ باشد را به پورت شماره‌ی ۲ می‌فرستد. این سیاست ^{۲۴} به سادگی رفتار سویچ‌ها را در نت‌کت تعریف می‌کند. در ادامه می‌توان با اضافه کردن یک فیلتر به این عبارت، ویژگی دسترسی کنترل ^{۲۵} را به این سیاست اضافه کرد تا همه‌ی بسته‌های از نوع SSH رها شوند:

$$p_{AC} \triangleq \neg(typ = SSH) \cdot p$$

اما استفاده از عبارت بالا به تنهایی برای توصیف رفتار شبکه شکل ۱.۱ کافی نیست. برای تکمیل این عبارت لازم است تا رفتار توپولوژی ^{۲۶} شبکه هم به آن افزوده شود. در نت‌کت توپولوژی شبکه به عنوان یک گراف جهت‌دار در نظر گرفته می‌شود و رفتار آن در قالب اجتماع رفتار هر یک از پیوندهای ^{۲۷} آن توصیف می‌شود. برای شبکه‌ی

^{۲۲}Switch^{۲۳}Host^{۲۴}Policy^{۲۵}Access Control^{۲۶}Topology^{۲۷}Link

شکل ۱.۱ می‌توان از عبارت زیر برای توصیف توپولوژی شبکه استفاده کرد:

$$\begin{aligned} t \triangleq & (sw = A \cdot pt = 2 \cdot sw \leftarrow B \cdot pt \leftarrow 1) + \\ & (sw = b \cdot pt = 1 \cdot sw \leftarrow A \cdot pt \leftarrow 2) + \\ & (sw = b \cdot pt = 2) \end{aligned}$$

در نکت در صورتی که سیاست و توپولوژی شبکه در قالب عبارت‌هایی توصیف شده باشند، رفتار کل شبکه در واقع دنباله‌ای از اعمال این عبارت‌ها به صورت یکی در میان است. به عنوان مثال در شکل ۱.۱ یک بسته از هاست ۱ ابتدا توسط سویچ A پردازش شده، سپس روی لینک بین دو سویچ جا به جا می‌شود و در نهایت توسط سویچ B پردازش می‌شود. در نکت می‌توان این رفتار را به صورت $p_{AC} \cdot t \cdot p_{AC}$ توصیف کرد. با استفاده از همین شهود، رفتار کل شبکه را می‌توان در قالب عبارت زیر توصیف کرد:

$$(p_{AC} \cdot t)^*$$

در توصیف بالا فرض شده است که بسته‌ها می‌توانند به هر طریق ممکن وارد شبکه و از آن خارج شوند، اما این رفتار همیشه مورد قبول نیست. به عنوان مثال در شبکه شکل ۱.۱ اگر مکان‌های ورودی و خروجی شبکه را در قالب عبارت زیر توصیف کنیم:

$$e \triangleq sw = A \cdot port = 1 \vee sw = B \cdot port = 2$$

می‌توانیم رفتار انتها به انتهای^{۲۸} شبکه را به شکل زیر توصیف کنیم:

$$p_{net} \triangleq e \cdot (p_{AC} \cdot t)^* e$$

در حالت کلی‌تر، نیازی به توصیف ورودی و خروجی‌های شبکه در قالب یک عبارت نیست. پس اگر فرض شود که مکان‌های ورودی شبکه توسط عبارت in و مکان‌های خروجی شبکه در قالب عبارت out توصیف شده باشند،

²⁸End to End

رفتار یک شبکه در نت کت به صورت زیر تعریف می شود:

$$in \cdot (p \cdot t)^* \cdot out$$

که عبارت p سیاست شبکه و عبارت t توپولوژی شبکه است.

۴.۳.۱ درستی سنجی برنامه‌های نت کت

درستی سنجی یک شبکه و بررسی خواص آن در نت کت با استفاده از بررسی تساوی عبارت یک شبکه با عبارت‌های دیگر انجام می شود. به عنوان مثال در شبکه‌ی شکل ۱.۱ برای بررسی اینکه همه‌ی بسته‌ها با نوع SSH از هاست ۱ رها می شوند کافی است تا تساوی زیر را ثابت کنیم:

$$\left(\begin{array}{l} type = SSH \cdot sw = A \cdot pt = 1 \cdot \\ (p_{AC} \cdot t)^* \cdot \\ sw = B \cdot pt = 2 \end{array} \right) \equiv 0$$

از طرفی برای بررسی یک خاصیت در شبکه، مثلاً امکان فرستاده شدن همه‌ی بسته‌هایی که از نوع SSH نیستند از هاست ۱ به هاست ۲ می توان به جای بررسی تساوی دو عبارت از نامساوی $p \leq q$ استفاده کرد. این نامساوی که خلاصه شده‌ی تساوی $p + q \equiv q$ است بیان می کند که رفتار عبارت p بخشی از رفتار عبارت q است. بنابراین برای بررسی این مساله که شبکه‌ی شکل ۱.۱ بسته‌های غیر SSH از هاست ۱ را عبور می دهد کافی است تا درستی نامعادله‌ی زیر را بررسی کرد:

$$\left(\begin{array}{l} \neg (type = SSH) \cdot sw = A \cdot pt = 1 \cdot \\ sw \leftarrow B \cdot pt \leftarrow 2 \end{array} \right) \leq (p_{AC} \cdot t)^*$$

۴.۱ نتکت پویا

نتکت پویا^{۲۹} برای رفع برخی از کاستی‌های نتکت ارائه شده است [۲]. به صورت دقیق‌تر نتکت پویا، امکان توصیف به‌روزرسانی سیاست‌های شبکه و همچنین رفتار شبکه در مقابل چندین بسته را ممکن می‌سازد.

۱.۴.۱ دستور زبان نتکت پویا

در نتکت پویا، از رفتار انتها به انتهای توصیف‌های شبکه در قالب عبارت‌های نتکت استفاده می‌شود. به همین منظور سینتکس نتکت پویا به صورت زیر تعریف می‌شود:

$$N ::= \text{NetKAT}^{-dup}$$

$$D ::= \perp | N; D | x?N; D | x!N; D | D \parallel DD \oplus D | X$$

$$X \triangleq D$$

در سینتکس بالا NetKAT^{-dup} قسمتی از زبان نتکت است که عبارت‌های dup از آن حذف شده است. عبارت‌های dup در توصیف‌های نتکت تأثیری در رفتار یک عبارت ندارند و هدف از استفاده از آن‌ها ثبت یک اثر از هر بسته پس از پردازش توسط یکی از عناصر شبکه است و امکان استدلال بر روی رفتار شبکه را ممکن می‌سازد. با توجه به این که در نتکت پویا رفتار انتها به انتهای یک عبارت نتکت مورد استفاده است، عبارت dup از دستور زبان کنار گذاشته شده است. نتکت پویا یک لیست از بسته‌های ورودی را پردازش می‌کند و یک لیست از مجموعه‌ی بسته‌های خروجی تولید می‌کند. اپراتور ترکیب متوالی^{۳۰} $N; D$ باعث می‌شود که یک بسته از لیست بسته‌های ورودی توسط سیاست N پردازش شود و سپس بسته‌ی توسط عبارت D پردازش شود. در نتکت پویا امکان ارتباط توسط عبارت‌هایی به شکل $x!N$ و $x?N$ توصیف می‌شوند که به ترتیب ارسال و دریافت یک عبارت نتکت را روی کانال x توصیف می‌کنند. ترکیب موازی^{۳۱} دو عبارت توسط $D \parallel D$ توصیف می‌شود. در نهایت رفتارهای غیرقطعی^{۳۲} توسط عبارت‌هایی به شکل $D \oplus D$ توصیف می‌شوند.

²⁹DyNetKAT

³⁰Sequential Composition

³¹Parallel Composition

³²Non-Deterministic

۲.۴.۱ معنای عملیاتی نتکت پویا

معنای عملیاتی^{۳۳} نتکت پویا با استفاده از عبارت‌هایی به شکل (d, H, H') تعریف می‌شوند که d عبارت نتکت پویا فعلی است، H لیست بسته‌هایی که در ادامه باید پردازش شوند و H' لیست بسته‌هایی است که به صورت موفقیت‌آمیز توسط شبکه پردازش شده‌اند. در اینجا فرض می‌شود که $F = \{f_1, \dots, f_n\}$ یک مجموعه از فیلدهای بسته‌ها است. یک بسته به شکل یک تابع $F \rightarrow \mathbb{N}$ توصیف می‌شود. برای یک بسته مانند σ تساوی $\sigma(f_i) = v_i$ بیان می‌کند که مقدار فیلد f_i در بسته‌ی σ برابر با v_i است. یک لیست خالی از بسته‌ها با $\langle \rangle$ نمایش داده می‌شود. اگر l یک لیست از بسته‌ها باشد $l :: e$ لیستی است که حاصل از اضافه کردن بسته σ به ابتدای لیست به دست می‌آید. برچسب هر قانون که با γ مشخص می‌شود به صورت یکی از شکل‌های $x!q, x?q, (\sigma, \sigma')$ یا $rcfg(x, q)$ تعریف می‌شود که $rcfg(x, q)$ به معنی انجام شدن $x!q$ و $x?q$ به صورت همگام^{۳۴} است. قوانین

^{۳۳}Operational Semantics^{۳۴}Synchronized

زیر معنای عملیاتی نت‌کت پویا را تعریف می‌کنند:

$$(cpol_{\rightarrow}^{\prime}) \frac{\sigma' \in \llbracket p \rrbracket (\sigma :: \langle \rangle)}{(p; q, \sigma :: H, H') \xrightarrow{(\sigma, \sigma')} (q, H, \sigma' :: H')} \quad (9.1)$$

$$(cpol_X) \frac{(p, H_0, H_1) \xrightarrow{\gamma} (p', H'_0, H'_1)}{(X, H_0, H_1) \xrightarrow{\gamma} (p', H'_0, H'_1)} X \triangleq p \quad (10.1)$$

$$(cpol_{\oplus}) \frac{(p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)}{(p \oplus q, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)} \quad (11.1)$$

$$(cpol_{\oplus-}) \frac{(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}{(p \oplus q, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)} \quad (12.1)$$

$$(cpol_{\parallel}) \frac{(p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)}{(p \parallel q, H_0, H'_0) \xrightarrow{\gamma} (p' \parallel q, H_1, H'_1)} \quad (13.1)$$

$$(cpol_{\parallel-}) \frac{(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}{(p \parallel q, H_0, H'_0) \xrightarrow{\gamma} (p \parallel q', H_1, H'_1)} \quad (14.1)$$

$$(cpol_{?}) \frac{}{(x?p; q, H, H') \xrightarrow{x?p} (q, H, H')} \quad (15.1)$$

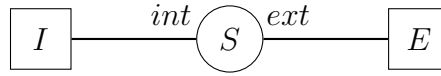
$$(cpol_{!}) \frac{}{(x!p; q, H, H') \xrightarrow{x!p} (q, H, H')} \quad (16.1)$$

$$(cpol_{!?}) \frac{(q, H, H') \xrightarrow{x!p} (q', H, H') (s, H, H') \xrightarrow{x?p} (s', H, H')}{(q \parallel, H, H') \xrightarrow{rcfg(x,p)} (q' \parallel s', H, H')} \quad (17.1)$$

$$(cpol_{?!}) \frac{(q, H, H') \xrightarrow{x?p} (q', H, H') (s, H, H') \xrightarrow{x!p} (s', H, H')}{(q \parallel, H, H') \xrightarrow{rcfg(x,p)} (q' \parallel s', H, H')} \quad (18.1)$$

قانون ۹.۱ انجام یک عملیات مانند (σ, σ') که به معنای پردازش بسته‌ی ابتدایی لیست ورودی توسط عبارت p و افزودن خروجی حاصل از آن مانند σ' به لیست خروجی است را مشخص می‌کند. قانون ۱۰.۱ بیان می‌کند که رفتار متغیر X که برابر با عبارت p است معادل با رفتار عبارت p است. قوانین ۱۱.۱ و ۱۲.۱ رفتار غیرقطعی را توصیف می‌کنند. قوانین ۱۳.۱ و ۱۴.۱ رفتار دو عبارت موازی را توصیف می‌کنند. قوانین ۱۵.۱ و ۱۶.۱ مشخص می‌کنند که ارسال یا دریافت پیام در نت‌کت پویا پردازشی روی بسته‌ها انجام نمی‌دهد. در نهایت همگام‌سازی^{۳۵} ارسال و دریافت پیام توسط قوانین ۱۵.۱ و ۱۶.۱ توصیف شده است.

³⁵Synchronization



شکل ۲.۱: مثال دیوار آتش

۳.۴.۱ توصیف برنامه‌ها در نکت پویا

در ادامه چگونگی توصیف یک دیوار آتش^{۳۶} وابسته به حالت^{۳۷} با استفاده از نکت پویا بیان می‌شود. شبکه‌ی شکل ۲.۱ را در نظر بگیرید. در این شبکه هدف این است که امکان ارتباط از داخل شبکه به بیرون فراهم باشد ولی امکان ارسال بسته از خارج شبکه ممکن نباشد. اما زمانی که یک بسته به خارج شبکه ارسال شد، دیوار آتش باید اجازه‌ی عبور بسته‌ها از بیرون را بدهد تا پاسخ بسته‌ها دریافت شوند. برای توصیف این شبکه می‌توان از عبارت نکت پویای زیر استفاده کرد:

$$Host \triangleq secConReq!1; Host \oplus secConEnd!1; Host$$

$$Switch \triangleq (port = int) \cdot (port \leftarrow ext); Switch \oplus$$

$$(port = ext) \cdot 0; Switch \oplus$$

$$secConReq?1; Switch'$$

$$Switch' \triangleq (port = int) \cdot (port \leftarrow ext); Switch' \oplus$$

$$(port = ext) \cdot (port \leftarrow int); Switch' \oplus$$

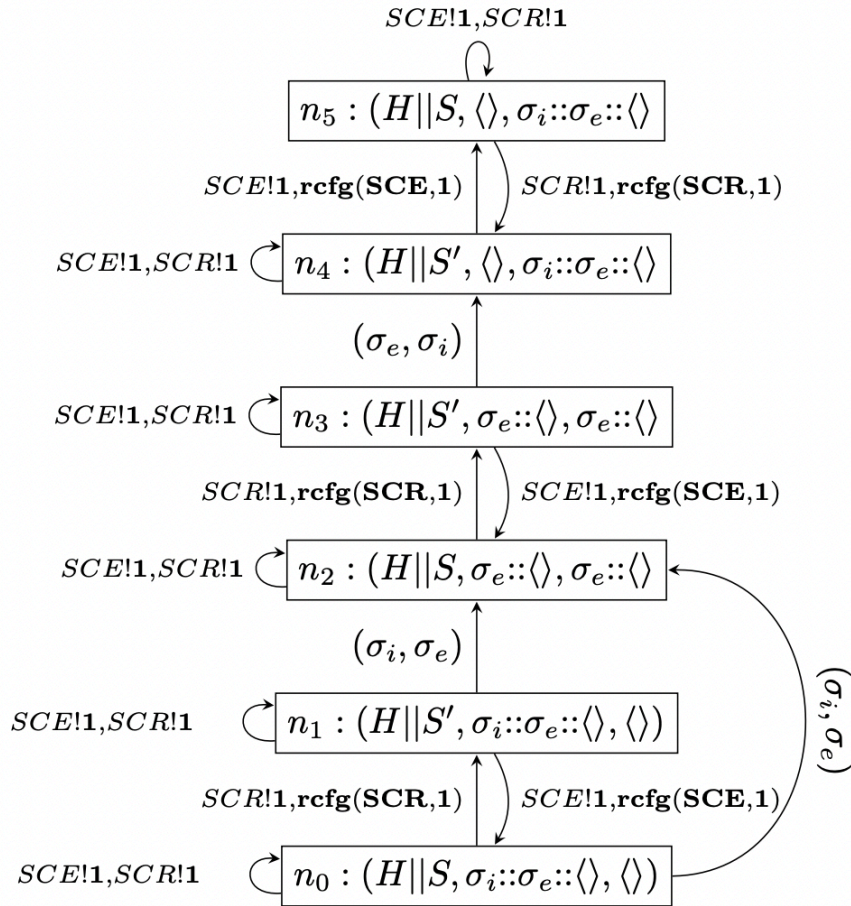
$$secConEnd?1; Switch$$

$$Init \triangleq Host \parallel Switch$$

در این توصیف عملیات $secConReq$ برای شروع ارتباط امن و $secConEnd$ برای خاتمه‌ی ارتباط امن در نظر گرفته شده‌است. بنابراین برنامه‌ی سوییچ پس از دریافت پیام برای شروع ارتباط امن تبدیل به برنامه‌ی $Switch'$ می‌شود که در آن اجازه‌ی ارسال بسته از پورت خارجی به پورت داخلی را می‌دهد. پس از دریافت پیام برای خاتمه‌ی ارتباط امن، برنامه به حالت اولیه‌ی خود بر می‌گردد و دوباره تمامی بسته‌های ورودی از پورت خارجی

³⁶Firewall³⁷Stateful

را رها می‌کند. در نهایت رفتار کل شبکه با استفاده از ترکیب موازی یک هاست و یک سویچ در حالت اولیه خود توصیف می‌شود. نمودار نمایش داده شده در شکل ۳.۱ سیستم انتقال برچسب‌دار^{۳۸} این شبکه را در حالتی



شکل ۳.۱: سیستم انتقال برچسب‌دار برای شبکه‌ی دیوار آتش

که یک بسته روی پورت ورودی و یک بسته روی پورت خروجی شبکه وجود دارد نشان می‌دهد. همانطور که در نمودار مشخص است، عملیات (σ_e, σ_i) که به معنای ارسال بسته از پورت ورودی به پورت خروجی است تنها در قسمتی از این سیستم انتقال قابل دسترسی است که پیش از آن یکی از عملیات‌های $SCR?1$ یا $rcfg(SCR, 1)$ انجام شده باشند. بنابراین در این حالت شبکه تنها در صورتی که بسته خارجی را به داخل ارسال می‌کند که پیش از آن پیام آغاز ارتباط امن دریافت کرده باشد.

³⁸Labeled Transition System

۵.۱ ساختمان رویداد

ساختمان رویداد ^{۳۹} [۷] یک مدل محاسباتی ^{۴۰} غیرجای‌گذاری شده ^{۴۱} برای پدازه‌های هم‌روند ^{۴۲} است. در این مدل، برخلاف مدل‌های جایگذاری شده ^{۴۳} مانند سیستم انتقال که هم‌روندی پدازه‌های موازی با انتخاب غیرقطعی مدل می‌شود، هم‌روندی پدازه‌ها به صورت صریح در مدل توصیف می‌شوند.

تعریف ۱.۵.۱. ساختمان رویداد یک ساختمان رویداد یک سه‌تایی $(E, \#, \vdash)$ است که در آن:

۱. E یک مجموعه از رویدادها است

۲. $\#$ رابطه‌ی تعارض ^{۴۴}، یک رابطه‌ی دودویی متقارن و غیربازتابی بر روی مجموعه‌ی E است

۳. $\vdash \subseteq Con \times E$ رابطه‌ی فعال‌سازی ^{۴۵} است که شرط زیر را برقرار می‌کند:

$$X \vdash e \wedge X \subseteq Y \in Con \Rightarrow Y \vdash e$$

در رابطه‌ی بالا Con زیرمجموعه‌ای از مجموعه‌ی توانی رویدادها است که اعضای آن فاقد تعارض باشند. به صورت دقیق‌تر داریم:

$$Con = \{X \subseteq E \mid \forall e, e' \in X. \neg(e \# e')\}$$

تعریف ۲.۵.۱. به ازای هر ساختمان رویداد، می‌توانیم رابطه‌ی فعال‌سازی مینیمال را به صورت زیر تعریف کنیم:

$$X \vdash_{min} e \iff X \vdash e \wedge (\forall Y \subseteq X. Y \vdash e \Rightarrow Y = X)$$

³⁹Event Structure

⁴⁰Computational Model

⁴¹Non-Interleaving

⁴²Concurrent

⁴³Interleaving

⁴⁴Conflict

⁴⁵Enabling

همچنین در هر ساختمان رویدادی شرط زیر برقرار است:

$$Y \vdash e \Rightarrow \exists X \subseteq Y. X \vdash_{min} e$$

برای مشخص کردن وضعیت یک سیستم در هر لحظه از مفهومی به نام پیکربندی^{۴۶} استفاده می‌شود و یک مجموعه شامل رویدادهایی است که تا آن لحظه در سیستم رخ داده‌اند.

تعریف ۳.۵.۱. اگر $E = (E, \#, \vdash)$ یک ساختمان رویداد باشد، یک پیکربندی آن یک زیرمجموعه از رویدادها $x \subseteq E$ است که شرایط زیر را داشته باشد:

$$1. x \in Con$$

$$2. \forall e \in x \exists e_0, \dots, e_n \in x. e_n = e \ \& \ \forall i \leq n. \{e_0, \dots, e_{i-1}\} \vdash e_i$$

مجموعه‌ی همه‌ی پیکربندی‌های یک ساختمان رویداد مانند E با $\mathcal{F}(E)$ نمایش داده می‌شود.

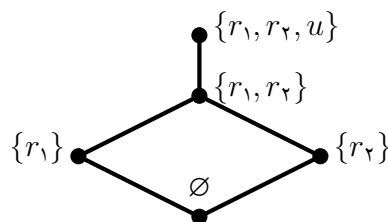
شبکه‌ی موجود در شکل ۵.۱ را در نظر بگیرید. در این شبکه دو هاست ۱ و ۲ به صورت هم‌رند یک بسته را به سویچ ارسال می‌کنند. این بسته‌ها شامل اطلاعات برای به روزرسانی مسیرهای دیگر در شبکه هستند، بنابراین سویچ پس از دریافت هر دوی این بسته‌ها آن‌ها را پردازش کرده و مسیرهای خود را به روزرسانی می‌کند. برای مدل کردن این شبکه می‌توانیم از یک ساختمان رویداد به صورت زیر استفاده کنیم:

$$E = (\{r_1, r_2, u\}, \emptyset, \{(\emptyset, r_1), (\emptyset, r_2), (\{r_1, r_2\}, u)\})$$

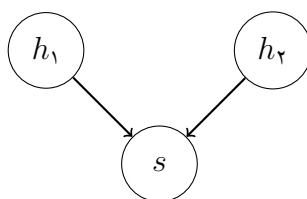
در این ساختمان رویداد، رویدادها به ترتیب دریافت یک بسته از هاست ۱، دریافت یک بسته از هاست ۲ و به روزرسانی سویچ را مدل می‌کنند. یکی از روش‌های رسم نمودار برای ساختمان رویداد، رسم نمودار هس^{۴۷} برای مجموعه‌ی پیکربندی‌های این ساختمان رویداد بر اساس رابطه‌ی زیرمجموعه است. برای مثالی که بیان شد می‌توان نموداری مطابق شکل ۴.۱ را رسم کرد.

⁴⁶ Configuration

⁴⁷ Hasse



شکل ۴.۱



شکل ۵.۱

۶.۱ مدل علی

پیدا کردن تعریفی برای علت واقعی^{۴۸} مبحثی است که مورد مطالعه و تحقیق بسیاری قرار گرفته است. این مساله به طور خاص در متون فلسفه مورد توجه قرار گرفته است. یکی از تعاریف علت واقعی که مورد توجه بسیاری قرار گرفته است، تعریفی مبتنی بر وابستگی خلاف واقع^{۴۹} است. مطابق این تعریف، رویداد الف علت رویداد ب است اگر در شرایطی که رویداد الف اتفاق نیافته باشد، رویداد ب هم اتفاق نیافتد. در اینجا اتفاق نیفتادن رویداد الف خلاف واقع است، چون در سناریوی واقعی (سناریو ای که واقعا اتفاق افتاده و مشاهده شده است) رویداد الف اتفاق افتاده است و در نظر گرفتن شرایطی که در آن رویداد الف اتفاق نیفتاده باشد بر خلاف واقعیت موجود است. اما این مدل به تنهایی امکان پیدا کردن علت مناسب را در همه‌ی موارد ندارد. به عنوان مثال سناریوی زیر را در نظر بگیرید که در آن سارا و بهرام هر کدام یک سنگ را برداشته و به سمت یک بطری شیشه‌ای پرتاب می‌کنند. در این سناریو، سنگ سارا زودتر از سنگ بهرام به بطری برخورد کرده و در نتیجه آن را می‌شکند. در این سناریو واضح است که پرتاب سنگ توسط سارا علت شکسته شدن بطری است. فرض کنید بخواهیم از علیت مبتنی بر خلاف واقع برای پیدا کردن این علت استفاده کنیم. بنابراین باید شرایطی را در نظر بگیریم که سارا سنگ خود را پرتاب نکند. اما مشکل اینجاست که در این شرایط همچنان بطری شکسته می‌شود، چون اگر سارا سنگ خود را پرتاب نکند، بهرام همچنان سنگ خود را پرتاب می‌کند و در نتیجه این بار سنگ بهرام

⁴⁸Actual Cause⁴⁹Counterfactual

به بطری برخورد کرده و آن را می‌شکند. بنابراین در این سناریو امکان تعریف پرتاب سنگ توسط سارا به عنوان علت شکسته شدن بطری با استفاده از استدلال مبتنی بر خلاف واقع وجود ندارد. هالپرن^{۵۰} و پزل^{۵۱} برای حل کردن مشکلاتی از این دست، تعریف جدیدی از علت واقعی [۴] ارائه کردند. مدل ارائه شده توسط آن‌ها به دلیل اینکه مبنای ریاضی دارد امکان استفاده از آن را در آنالیز و تحلیل سیستم‌های محاسباتی فراهم می‌کند. به همین دلیل این تعریف در مقالات زیادی در حوزه‌ی دانش کامپیوتر مورد استفاده قرار گرفته است. برای تعریف علت واقعی ابتدا برخی مفاهیم اولیه مورد استفاده در این تعریف توضیح داده می‌شوند. به صورت کلی فرض می‌شود که دنیای مورد تحلیل توسط تعدادی متغیر تصادفی مدل شده است. اگر X یک متغیر تصادفی باشد، یک رویداد به شکل $X = x$ تعریف می‌شود. برخی از این متغیرها بر روی یکدیگر تاثیر گذارند. این وابستگی‌ها در قالب مجموعه‌ای از معادلات ساختاری^{۵۲} مدل می‌شوند. هر یک از این معادلات در واقع یک مکانیزم یا قانون مشخص در این دنیا را مدل می‌کنند. متغیرها به دو دسته درونی^{۵۳} و برونی^{۵۴} تقسیم می‌شوند. متغیرهای برونی متغیرهایی در نظر گرفته می‌شوند که مقدار آن‌ها توسط عواملی که درون مدل نیستند تعیین می‌شوند. بنابراین در مدل فرض می‌شود که مقدار این متغیرها از قبل مشخص است. اما متغیرهای درونی متغیرهایی هستند که مقدار آن‌ها بر اساس معادلات ساختاری تعیین می‌شود. به صورت دقیق‌تر، امضای^{۵۵} یک مدل یک سه‌تایی $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$ است که در آن \mathcal{U} مجموعه‌ی متغیرهای بیرونی \mathcal{V} مجموعه‌ی متغیرهای درونی و \mathcal{R} دامنه‌ی مقادیر ممکن برای هر یک از متغیرها را مشخص می‌کند. در این مدل فرض می‌شود که مجموعه‌ی متغیرهای درونی محدود است. مدل علی بر روی یک امضای \mathcal{S} یک دوتایی $\mathcal{M} = (\mathcal{S}, \mathcal{F})$ است که در آن \mathcal{F} به هر متغیر داخلی $X \in \mathcal{V}$ یک تابع $F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U)) \times (\times_{Y \in \mathcal{V} - \{X\}} \mathcal{R}(Y)) \rightarrow \mathcal{R}(X)$ اختصاص می‌دهد. هر تابع، معادله‌ی یک متغیر را به ازای مقادیر تمام متغیرهای دیگر مشخص می‌کند. به عنوان مثال اگر فرض کنیم $F_X(Y, Z, U) = Y + U$ اگر داشته باشیم $U = 2$, $Y = 3$ آنگاه مقدار X برابر ۵ خواهد شد. این معادلات امکان تفسیر آن‌ها بر اساس شرایط خلاف واقع را می‌دهند. به عنوان مثال در همین مدل اگر فرض کنیم که $U = u$ می‌توانیم نتیجه بگیریم که اگر مقدار متغیر Y برابر ۴ باشد آنگاه مستقل از اینکه مقدار بقیه‌ی متغیرها در دنیای واقعی چه مقداری دارند، مقدار متغیر X برابر $u + 4$ خواهد بود که به صورت $(M, u) \models [Y \leftarrow 4](X = u + 4)$ نوشته می‌شود. توابع ذکر

⁵⁰Halpern⁵¹Pearl⁵²Structural Equations⁵³Endogenous⁵⁴Exogenous⁵⁵Signature

شده فقط برای متغیرهای درونی تعریف می‌شوند و همانطور که پیش‌تر اشاره شد، برای متغیرهای بیرونی تابعی تعریف نمی‌شود و فرض می‌شود که مقدار آن‌ها از قبل مشخص شده است.

مثال ۱.۶.۱. یک جنگل را در نظر بگیرید که می‌تواند توسط رعد و برق یا یک کبریت رها شده دچار آتش سوزی شود. برای مدل کردن این سناریو از سه متغیر بولی^{۵۶} استفاده می‌کنیم:

- متغیر F که اگر جنگل دچار آتش سوزی شود مقدار آن درست است و در غیر این صورت مقدار آن غلط است

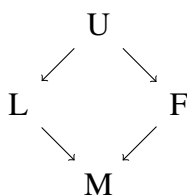
- متغیر L که اگر رعد و برق اتفاق افتاده باشد مقدار آن درست است و در غیر این صورت غلط است

- متغیر M که اگر یک کبریت در جنگل رها شده باشد مقدار آن درست است و در غیر این صورت غلط است

در این مثال فرض می‌کنیم که مقادیر متغیرهای برونی به گونه‌ای است که تمام شرایط لازم برای آتش سوزی جنگل در صورتی که رعد و برق اتفاق بیافتد یا کبریتی در جنگل رها شود را دارد (به عنوان مثال درختان جنگل به اندازه‌ی کافی خشک هستند و اکسیژن کافی در هوا وجود دارد). در این مدل تابع متغیر F را به گونه‌ای تعریف می‌کنیم که داشته باشیم: $F_F(\vec{u}, L, M) = L \vee M$. همانطور که پیش‌تر بیان شد، این مدل علی امکان بررسی معادلات بر اساس شرایط خلاف واقع را می‌دهد. به صورت دقیق‌تر اگر $M = (\mathcal{S}, \mathcal{F})$ یک مدل علی، \vec{X} یک بردار از متغیرهای درونی و \vec{x}, \vec{u} برداری از مقادیر متغیرهای \mathcal{U}, \vec{X} باشند مدل $M_{\vec{X} \leftarrow \vec{x}}$ را با امضای $S_{\vec{X}}$ $(\mathcal{U}, \mathcal{V} - \vec{X}, \mathcal{R}|_{-\vec{X}})$ یک زیرمدل^{۵۷} از M تعریف می‌کنیم. به صورت شهودی این مدل حاصل مداخله^{۵۸} ای در مدل M است که در آن مقادیر \vec{x} را به متغیرهای \vec{X} اختصاص داده‌ایم. به صورت دقیق‌تر تعریف می‌کنیم $M_{\vec{X} \leftarrow \vec{x}} = (\mathcal{S}_{\vec{X}}, \mathcal{F}^{\vec{X} \leftarrow \vec{x}})$ که $F_Y^{\vec{X} \leftarrow \vec{x}}$ از تابع F_Y که در آن مقادیر \vec{x} را به متغیرهای \vec{X} اختصاص داده‌ایم به دست می‌آید. به عنوان مثال اگر M مدل مثال ۱.۶.۱ باشد آنگاه در مدل $M_{L \leftarrow F}$ معادله‌ی متغیر F به $F = M$ تبدیل می‌شود. این معادله دیگر به متغیر L وابسته نیست بلکه با توجه به مقدار آن که در اینجا غلط است معادله‌ی جدیدی دارد. علاوه بر این توجه کنید که در مدل $M_{L \leftarrow F}$ دیگر معادله‌ای برای متغیر L وجود ندارد. توجه کنید که در حالت کلی ممکن است یک بردار یکتا از مقادیر متغیرها برای یک مدل وجود نداشته باشد که همزمان تمامی

^{۵۶}Boolean^{۵۷}Sub-Model^{۵۸}Intervention

معادلات را حل کند. در مدل علی یک بردار از مقادیر متغیرهای برونی \vec{u} یک هم‌بافت^{۵۹} نامیده می‌شود. در مدل‌های بازگشتی به ازای یک هم‌بافت مشخص همیشه یک راه‌حل یکتا برای تمامی معادلات مدل وجود دارد. در ادامه فرض می‌شود که مدل‌ها بازگشتی هستند. تعمیم مدل علی برای مدل‌های غیربازگشتی در [۴] توضیح داده است. برای یک مدل می‌توان یک شبکه‌ی علی ترسیم کرد. این شبکه یک گراف جهت‌دار است که به ازای هر متغیر یک گره در آن وجود دارد و یک یال بین دو گره وجود دارد اگر تابع متغیر دوم به متغیر اول وابسته باشد. به عنوان مثال شکل زیر شبکه‌ی علی مثال ۱.۶.۱ را نشان می‌دهد:



در ادامه برای سادگی رسم شبکه‌ی علی، متغیرهای برونی را از آن‌ها حذف می‌کنیم.

۱.۶.۱ علت واقعی

در ادامه فرمول‌های لازم برای تعریف علت واقعی توصیف می‌شوند. اگر $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$ یک امضا باشد فرمول $X = x$ یک رویداد بدوی^{۶۰} نامیده می‌شود که $X \in \mathcal{V}, x \in \mathcal{R}(X)$. فرمول $[Y_1 \leftarrow y_1, \dots, Y_k \leftarrow y_k] \varphi$ یک فرمول علی پایه^{۶۱} نامیده می‌شود که در آن:

- φ یک ترکیب بولی از رویدادهای بدوی است

- Y_1, \dots, Y_k متغیرهای متمایز در \mathcal{V} هستند

- $y_i \in \mathcal{R}(Y_i)$

این فرمول به صورت خلاصه به شکل $[\vec{Y} \leftarrow \vec{y}] \varphi$ نوشته می‌شود و اگر $k = 0$ باشد آنگاه به صورت φ نوشته می‌شود. به صورت شهودی یک فرمول به شکل $[\vec{Y} \leftarrow \vec{y}] \varphi$ بیان می‌کند که در شرایط خلاف واقع ای که در آن

⁵⁹ Context

⁶⁰ Prime Event

⁶¹ Basic Causal Formula

مقادیر \vec{y} به متغیرهای \vec{Y} اختصاص داده شده است فرمول φ برقرار است. یک فرمول علی به صورت یک ترکیب بولی از فرمول‌های علی پایه تعریف می‌شود. برقراری فرمول علی ψ در مدل M تحت هم‌بافت \vec{u} را به صورت $\psi(M, \vec{u}) \models \psi$ نشان می‌دهیم. به عنوان مثال $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}](X = x)$ برقرار است اگر مقدار متغیر X در راه حل معادلات مدل $M_{\vec{Y} \leftarrow \vec{y}}$ تحت هم‌بافت \vec{u} برابر x باشد.

تعریف ۲.۶.۱. فرمول $\vec{X} = \vec{x}$ علت واقعی φ (که تاثیر^{۶۲} نامیده می‌شود) در (M, \vec{u}) است اگر شرایط زیر برای آن برقرار باشد:

$$1. (M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \varphi$$

۲. یک افزاز مانند (\vec{Z}, \vec{W}) از مجموعه متغیرهای \mathcal{V} با شرط $\vec{X} \subseteq \vec{Z}$ و مقادیر (\vec{x}, \vec{w}') برای متغیرهای (\vec{X}, \vec{W}) وجود داشته باشد که داشته باشیم $(M, \vec{u}) \models \vec{Z} = \vec{z}^*$ و شرایط زیر را برآورده کند:

$$(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W} \leftarrow \vec{w}'] \neg \varphi \quad (1)$$

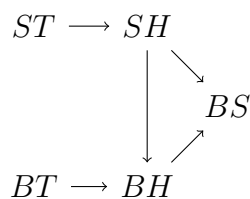
$$\forall \vec{W}' \subseteq \vec{W}, \vec{Z}' \in \vec{Z}. (M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*] \varphi \quad (ب)$$

۳. \vec{X} مینیمال باشد.

در این تعریف شرط اول بیان می‌کند که علت و تاثیر هر دو در شرایط واقعی برقرار هستند. شرط دوم به دنبال پیدا کردن شرایطی است که تحت آن تاثیر به صورت غیر واقع به علت وابسته باشد. این شرایط متغیرهای \vec{W} و مقادیری مانند \vec{w}' برای آن‌ها هستند. شرط ۲. الف بررسی می‌کند که تحت شرایطی که توسط $\vec{W} \leftarrow \vec{w}'$ به وجود می‌آید اگر علت مقداری متفاوت از مقدار خود در هم‌بافت واقعی داشته باشد اثر در مدل دیده نمی‌شود. شرط ۲. ب بررسی می‌کند که شرایط استفاده شده در ۲. الف عامل از بین رفتن اثر در ۲. الف نباشند. برای این منظور در شرایطی که علت مقدار واقعی خود را دارد در تمامی حالت‌هایی که متغیرهای شرایط می‌توانند داشته باشند بررسی می‌شود که اثر همچنان برقرار باشد. شرط سوم در واقع بیان می‌کند که زیرمجموعه‌ای از علت وجود نداشته باشد که همزمان شرایط ۱ و ۲ را برقرار کند. در تعریف بالا $(\vec{W}, \vec{w}', \vec{x}')$ یک شاهد^{۶۳} بر اینکه $\vec{X} = \vec{x}$ علت φ است تعریف می‌شود.

⁶²Effect

⁶³Witness



شکل ۶.۱

۲.۶.۱ پیدا کردن علت واقعی در مسائل

در ادامه مثال سارا و بهرام که در ابتدای این بخش ذکر شده بود را بررسی می‌کنیم.

برای مدل کردن این مساله متغیرهای زیر را در نظر می‌گیریم:

- BT : پرتاب سنگ توسط بهرام
- BH : برخورد سنگ بهرام به بطری
- ST : پرتاب سنگ توسط سارا
- SH : برخورد سنگ سارا به بطری
- BS : شکسته شدن بطری

ابتدا فرض می‌کنیم که متغیرهای BT , ST تنها به متغیرهای برونی وابسته‌اند. بطری در صورتی شکسته می‌شود که هر یک از سنگ‌های سارا یا بهرام با آن برخورد کنند. بنابراین برای شکسته شدن بطری معادله‌ی $BS = SH \vee BH$ را در نظر می‌گیریم. نکته‌ی اصلی در این مساله این است که سنگ سارا زودتر از سنگ بهرام به شیشه برخورد می‌کند، به همین دلیل لازم است تا این موضوع در مدل لحاظ شود. یک راه برای مدل کردن این مساله این است که معادله‌ی برخورد سنگ بهرام به شیشه را به گونه‌ای تعریف کنیم که تنها در صورتی که سنگ سارا به بطری برخورد نکرده باشد آنگاه سنگ بهرام به بطری برخورد کند. بنابراین می‌توانیم معادله‌ی $BH = BT \wedge \neg SH$ را تعریف کنیم. علاوه بر این معادله‌ی برخورد سنگ سارا را بدون وابستگی به برخورد سنگ بهرام تعریف می‌کنیم: $SH = ST$. با توجه به این تعاریف برای معادلات می‌توانیم گراف علی شکل ۶.۱ را برای این مدل رسم کنیم. در این مدل می‌توانیم $ST = T$ را به عنوان علت $BS = T$ تعریف کنیم. برای برقراری شرط ۲ در تعریف علت واقعی شرایط $\{BT\}$ و $\vec{W} = F$ و $w' = F$ را در نظر می‌گیریم. در این شرایط چون مقدار BH برابر F می‌شود،

مقدار BS تنها وابسته به مقدار SH و در نتیجه ST می شود. همچنین در این مدل $BT = T$ علت شکسته شدن شیشه نیست. مثلاً فرض کنید که شرایط $w' = F$, $\bar{W} = \{ST\}$ را در نظر بگیریم. در این شرایط اگر مقدار BT را به F تغییر دهیم مقدار BS هم غلط می شود. بنابراین شرط ۲. الف برقرار است. اما به ازای $\bar{Z}' = \{BH\}$ شرط ۲. ب برقرار نمی شود. در این حالت داریم: $(M, \bar{u}) \models [BT \leftarrow T, ST \leftarrow F, BH \leftarrow F]BS = F$ توجه کنید با وجود اینکه مقدار درست به BT اختصاص یافته اما چون مقدار BH به مقدار آن در هم بافت واقعی برگردانده می شود در نتیجه مقدار BS همچنان غلط می ماند.

مثال بالا نشان می دهد که این تعریف از علت واقعی برخی از مشکلات موجود در تعاریف ساده مبتنی بر خلاف واقع را برطرف می کند و می تواند توضیح مناسبی در برخی از این مثال ها پیدا کند. نکته ای که باید به آن توجه شود این است که هنوز روش یا معیاری برای این که چه تعریفی از علت واقعی تعریف مناسب است وجود ندارد. تنها روش ممکن مقایسه تعاریف مختلف استفاده از آن ها در مساله ها و سناریوهای مختلف و بررسی تطابق علت به دست آمده با استفاده از این تعریف ها با شهود موجود از مساله است.

۳.۶.۱ مدل تعمیم یافته

مدل علی تعمیم یافته^{۶۴} یک سه تایی $(S, \mathcal{F}, \mathcal{E})$ است که (S, \mathcal{F}) یک مدل علی است و \mathcal{E} یک مجموعه از مقداردهی های مجاز^{۶۵} برای متغیرهای درونی است. به صورت دقیق تر اگر متغیرهای درونی X_1, \dots, X_n باشند آن گاه $(x_1, \dots, x_n) \in \mathcal{E}$ اگر $X_1 = x_1, \dots, X_n = x_n$ یک مقداردهی مجاز است. یک مقداردهی دلخواه به یک زیرمجموعه از متغیرهای درونی مجاز است اگر امکان تعمیم به یک مقداردهی مجاز در \mathcal{E} را داشته باشد. هدف از این تعریف جلوگیری از در نظر گرفتن علت هایی است که شرایط رخ دادن آن ها غیر محتمل است. با توجه به تعریف مقداردهی مجاز، علت واقعی در یک مدل تعمیم یافته به گونه ای تعریف می شود که در شرط ۲ فقط امکان مقداردهی های مجاز وجود داشته باشد. در [۴] تعریف دقیق علت واقعی در مدل تعمیم یافته بیان نشده است. در بخش بعدی تعریفی از علت واقعی در مدل تعمیم یافته ارائه می شود.

⁶⁴Extended Causal Model

⁶⁵Allowable Settings

فصل ۲

روش و راه حل پیشنهادی

۱.۲ مقدمه

در این فصل روش پیدا کردن علت خطا در شبکه‌های نرم‌افزاری توضیح داده می‌شود. در بخش اول معنای عبارات نت‌کت پویا با استفاده از ساختمان رویداد تعریف می‌شود. در بخش دوم یک مدل علی برای توصیف ساختمان رویداد مطرح می‌شود. در نهایت بخش سوم شامل استفاده از این روش‌ها برای توضیح خطا در شبکه‌های نرم‌افزاری با استفاده از چند مثال بیان می‌شود.

۲.۲ معنای عبارات نت‌کت پویا در قالب ساختمان رویداد

در این بخش ابتدا شیوه‌ی اعمال چند نوع عملیات برای ترکیب ساختمان‌های رویداد تعریف می‌شود. سپس با استفاده از این عملیات‌ها معنای عبارات نت‌کت پویا توصیف می‌شود که برگرفته از [۷] می‌باشد.

تعریف ۱.۲.۲. محدودیت فرض کنید $E = (E, \#, \vdash)$ یک ساختمان رویداد باشد. به ازای یک مجموعه‌ی $A \subseteq E$ محدودیت E ^۱ به A یک ساختمان رویداد به صورت زیر است:

$$E[A] = (A, \#_A, \vdash_A)$$

که داشته باشیم:

$$X \subseteq \text{Con}_A \iff X \subseteq A \wedge X \in \text{Con}$$

$$X \vdash_A e \iff X \subseteq A \wedge e \in A \wedge X \vdash e$$

تعریف ۲.۲.۲. فرض کنید $E = (E, \#, \vdash)$ یک ساختمان رویداد و a یک رویداد باشد. ساختمان رویداد $aE = (E', \#', \vdash')$ به گونه‌ای تعریف می‌شود که داشته باشیم:

$$E' = \{(0, a)\} \cup \{(1, e) | e \in E\},$$

$$e'_0 \# e'_1 \iff \exists e_0, e_1. e'_0 = (1, e_0) \wedge e'_1 = (1, e_1) \wedge e_0 \# e_1$$

$$X \vdash' e' \iff e' = (0, a) \text{ or } [e' = (1, e_1) \wedge (0, a) \in X \wedge \{e | (1, e) \in X\} \vdash e_1]$$

تعریف ۳.۲.۲. ساختمان رویداد برچسب‌دار یک ساختمان رویداد برچسب‌دار^۲ یک پنج‌تایی به شکل $(E, \#, \vdash, L, l)$ است که در آن $(E, \#, \vdash)$ یک ساختمان رویداد است، L یک مجموعه از برچسب‌ها (فاقد عنصر $*$) و l یک تابع به فرم $l : E \rightarrow L$ است که به هر رویداد یک برچسب اختصاص می‌دهد. یک ساختمان رویداد برچسب‌دار را به اختصار به صورت (E, L, l) نشان می‌دهیم.

تعریف ۴.۲.۲. در یک ساختمان رویداد رابطه‌ی \bowtie را به صورت زیر تعریف می‌کنیم:

$$e \bowtie e' \iff e \# e' \vee e = e'$$

¹Restriction

²Labeled Event Structure

۱.۲.۲ ترکیب ساختمان رویدادهای برچسب‌دار

تعریف ۵.۲.۲. فرض کنید (E, L, l) یک ساختمان رویداد برچسب‌دار باشد و α یک برچسب باشد. $\alpha(E, L, l)$

را به صورت یک ساختمان رویداد برچسب‌دار به شکل $(\alpha E, L', l)$ تعریف می‌کنیم که در آن $L' = \{\alpha\} \cup L$ و به ازای هر $e' \in E'$ داریم:

$$l'(e') = \begin{cases} \alpha & \text{if } e = (0, \alpha) \\ l(e) & \text{if } e = (1, e) \end{cases}$$

تعریف ۶.۲.۲. فرض کنید $E_0 = (E_0, \#_0, \vdash_0, L_0, l_0)$ و $E_1 = (E_1, \#_1, \vdash_1, L_1, l_1)$ دو ساختمان رویداد برچسب‌دار باشند. مجموع این دو ساختمان رویداد $E_0 + E_1$ را به صورت یک ساختمان رویداد برچسب‌دار $(E, \#, \vdash, L, l)$ تعریف می‌کنیم که در آن داشته باشیم:

$$E = \{(0, e) | e \in E_0\} \cup \{(1, e) | e \in E_1\}$$

با استفاده از این مجموعه از رویدادها توابع $\iota_k : E_k \rightarrow E$ را به صورت زیر و به ازای $k = 0, 1$ تعریف می‌کنیم:

$$\iota_k(e) = (k, e)$$

رابطه‌ی تعارض را به گونه‌ای تعریف می‌کنیم که داشته باشیم:

$$\begin{aligned} e \# e' &\iff \exists e_0, e'_0. e = \iota_0(e_0) \wedge e' = \iota_0(e'_0) \wedge e_0 \#_0 e'_0 \\ &\vee \exists e_1, e'_1. e = \iota_1(e_1) \wedge e' = \iota_1(e'_1) \wedge e_1 \#_1 e'_1 \\ &\vee \exists e_0, e_1. (e = \iota_1(e_0) \wedge e' = \iota_1(e_1)) \vee (e' = \iota_1(e_0) \wedge e = \iota_1(e_1)) \end{aligned}$$

رابطه ی فعال سازی را به گونه ای تعریف می کنیم که داشته باشیم:

$$X \vdash e \iff X \in Con \wedge e \in E \wedge$$

$$(\exists X_0 \in Con_0, e_0 \in E_0. X = \iota_0 X_0 \wedge e = \iota_0(e_0) \wedge X_0 \vdash_0 e_0) \text{ or}$$

$$(\exists X_1 \in Con_1, e_1 \in E_1. X = \iota_1 X_1 \wedge e = \iota_1(e_1) \wedge X_1 \vdash_1 e_1)$$

و مجموعه ی برچسب ها را به صورت $L = L_0 \cup L_1$ و تابع برچسب گذاری را به صورت زیر تعریف می کنیم:

$$l(e) = \begin{cases} l_0(e_0) & \text{if } e = \iota_0(e_0) \\ l_1(e_1) & \text{if } e = \iota_1(e_1) \end{cases}$$

تعریف ۷.۲.۲. فرض کنید که $E_0 = (E_0, \#_0, \vdash_0, L_0, l_0)$ و $E_1 = (E_1, \#_1, \vdash_1, L_1, l_1)$ دو ساختار رویداد برچسب گذاری شده باشند. حاصل ضرب آن ها $E_0 \times E_1$ را به صورت یک ساختمان رویداد برچسب گذاری شده $E = (E, \#, \vdash, L, l)$ تعریف می کنیم که در آن رویدادها به صورت زیر تعریف می شوند:

$$E_0 \times_* E_1 = \{(e_0, *) | e_0 \in E_0\} \cup \{(*, e_1) | e_1 \in E_1\} \cup \{(e_0, e_1) | e_0 \in E_0 \wedge e_1 \in E_1\}$$

با توجه به این مجموعه رویدادها توابعی به شکل $\pi_i : E \rightarrow_* E_i$ تعریف می کنیم که به ازای $i = 0, 1$ داشته باشیم: $\pi_i(e_0, e_1) = e_i$. در اینجا رابطه ی تعارض را به کمک رابطه ی \bowtie که پیش تر تعریف شد، به شکل زیر به ازای تمامی رویدادهای $e, e' \in E$ توصیف می کنیم:

$$e \bowtie e' \iff \pi_0(e) \bowtie_0 \pi_0(e') \vee \pi_1(e) \bowtie_1 \pi_1(e')$$

رابطه‌ی فعال‌سازی را به صورت زیر تعریف می‌کنیم:

$$X \vdash e \iff X \in Con \wedge e \in \mathcal{E}$$

$$(\pi_0(e) \text{ defined} \Rightarrow \pi_0 X \vdash_0 \pi_0(e)) \wedge (\pi_1(e) \text{ defined} \Rightarrow \pi_1 X \vdash_1 \pi_1(e))$$

مجموعه‌ی برچسب‌های حاصلضرب را به صورت زیر تعریف می‌کنیم:

$$L_0 \times_* L_1 = \{(\alpha_0, *) | \alpha_0 \in L_0\} \cup \{(*, \alpha_1) | \alpha_1 \in L_1\} \cup \{(\alpha_0, \alpha_1) | \alpha_0 \in L_0 \wedge \alpha_1 \in L_1\}$$

در انتها تابع برچسب‌گذاری را به صورت زیر تعریف می‌کنیم:

$$l(e) = (l_0(\pi_0(e), l_1(\pi_1(e))))$$

تعریف ۸.۲.۲. فرض کنید که $E = (E, \#, \vdash, L, l)$ یک ساختمان رویداد برچسب‌دار باشد. فرض کنید Λ یک زیرمجموعه از L باشد. محدودیت E به Λ را به صورت $E[\Lambda]$ و به شکل یک ساختمان رویداد برچسب‌گذاری شده به شکل $(E', \#', \vdash', L \cap \Lambda, l')$ که در آن $\{e \in E | l(e) \in \Lambda\}$ است و $(E', \#', \vdash') = (E, \#, \vdash)[\{e \in E | l(e) \in \Lambda\}]$ است و تابع برچسب‌گذاری معادل محدودسازی تابع l به دامنه‌ی $L \cap \Lambda$ است.

۲.۲.۲ معنای عبارات نت‌کت پویا در قالب ساختمان رویداد برچسب‌دار

در ادامه چگونگی تعریف معنا^۳ برای عبارات‌های نت‌کت پویا بیان می‌شود. در نت‌کت پویا رفتار عبارات‌های نت‌کت فقط به صورت انتها به انتها در نظر گرفته می‌شوند. با توجه به همین موضوع در ادامه قسمتی از نت‌کت پویا مورد استفاده قرار می‌گیرد که عبارات‌های نت‌کت به صورت نرمال در آن ظاهر می‌شوند. اگر فرض کنیم که مجموعه‌ی فیلدها f_1, f_2, \dots, f_k باشد یک فیلتر کامل^۴ به صورت $\alpha = f_1 = n_1 \dots f_k = n_k$ و یک اختصاص کامل^۵ به صورت $\pi = f_1 \leftarrow n_1 \dots f_k \leftarrow n_k$ تعریف می‌شود. می‌گوییم یک عبارت q در $NetKAT^{dup}$ به

³Semantic

⁴Complete Test

⁵Complete Assignment

فرم نرمال است اگر به شکل $\sigma_{\alpha \cdot \pi \in \mathcal{A}} \alpha \cdot \pi$ باشد که داشته باشیم $\mathcal{A} = \{\alpha_i \cdot \pi_i | i \in I\}$. در عبارت قبل I مدل زبانی $NetKAT^{dup}$ می باشد. بر اساس لم ۵ در [۲] به ازای هر عبارت p در $NetKAT^{dup}$ یک عبارت p' به فرم نرمال وجود دارد که داشته باشیم: $p \equiv p'$. در ادامه از بخشی از زبان نتکت پویا که آن را نتکت پویا نرمال می نامیم برای ارائه معنا در قالب ساختمان رویداد استفاده می کنیم.

تعریف ۹.۲.۲. زبان نتکت پویا نرمال را با دستور زبان زیر تعریف می کنیم:

$$F ::= \alpha \cdot \pi$$

$$D ::= \perp | F; D | x?F; D | x!F; D | D \parallel D | D \oplus D$$

بر اساس لم ۹ در [۲] به ازای هر عبارت p در نتکت پویا یک عبارت معادل آن در نتکت پویا نرمال وجود دارد که داشته باشیم: $p \equiv q$. در ادامه فرض کنیم که \mathcal{A} مجموعه ای الفبا شامل تمامی حروف به شکل $\alpha \cdot \pi, x?F, x!F$ باشد و داشته باشیم $\alpha \in \mathcal{A}$. معنای عبارت های بر روی نتکت پویای نرمال را به صورت زیر تعریف می کنیم:

$$\llbracket \perp \rrbracket = (\emptyset, \emptyset)$$

$$\llbracket \alpha; t \rrbracket = \alpha(\llbracket t \rrbracket)$$

$$\llbracket t_1 \oplus t_2 \rrbracket = \llbracket t_1 \rrbracket + \llbracket t_2 \rrbracket$$

$$\llbracket t_1 \parallel t_2 \rrbracket = \llbracket t_1 \rrbracket \times \llbracket t_2 \rrbracket$$

$$\llbracket \delta_L(t) \rrbracket = \llbracket t \rrbracket[\mathcal{A} \setminus L]$$

۳.۲ مدل علی برای ساختمان رویداد

در ادامه نحوه ی توصیف یک ساختمان رویداد در قالب یک مدل علی را بیان می کنیم. فرض کنیم که $E = (E, \#, \vdash)$ یک ساختمان رویداد باشد. مدل علی این ساختمان رویداد را به صورت $\mathcal{M} = (\mathcal{F}, \mathcal{E})$ تعریف می کنیم که در آن $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$. در این مدل فرض می کنیم همه ی متغیرها از نوع

بولی هستند. همچنین در این مدل متغیر برونی در نظر نمی‌گیریم بنابراین داریم $\mathcal{U} = \emptyset$. اگر فرض کنیم مجموعه رویدادها به صورت $E = \{e_1, e_2, \dots, e_n\}$ باشد مجموعه‌ی متغیرهای درونی را به صورت زیر تعریف می‌کنیم:

$$\begin{aligned} \mathcal{V} = & \{C_{e_i, e_j} | 1 \leq i < j \leq n, e_i \in E \wedge e_j \in E\} \\ & \cup \{EN_{s, e} | s \in \mathcal{P}(E), e \in E, e \notin s\} \\ & \cup \{M_{s, e} | s \in \mathcal{P}(E), e \in E, e \notin s\} \cup \{PV\} \end{aligned}$$

به صورت شهودی به ازای هر عضو از رابطه‌های $\#, \vdash, \vdash_{min}$ یک متغیر درونی در نظر می‌گیریم که درست بودن این متغیر به معنای وجود عنصر متناظر با آن در رابطه است.

به ازای $x, y \in \mathcal{P}(E)$ پوشیده شدن^۶ x توسط y را که با $x < y$ نمایش می‌دهیم به صورت زیر تعریف می‌کنیم:

$$x \subseteq y \wedge x \neq y \wedge (\forall z. x \subseteq z \subseteq y \Rightarrow x = z \text{ or } y = z)$$

همچنین به ازای هر متغیر $X \in \mathcal{V}$ بردار \vec{V}_X را بردار شامل همه‌ی متغیرهای درونی به غیر از X تعریف می‌کنیم. با استفاده از این تعاریف توابع متغیرهای درونی را به صورت زیر تعریف می‌کنیم:

$$F_{C_{e, e'}}(\vec{V}_{C_{e, e'}}) = \begin{cases} true & \text{if } e \# e' \\ false & \text{otherwise} \end{cases}$$

$$F_{M_{s, e}}(\vec{V}_{M_{s, e}}) = \begin{cases} Min(s, e) \wedge Con(s) & \text{if } s \vdash_{min} e \\ false & \text{otherwise} \end{cases}$$

$$F_{EN_{s, e}}(\vec{V}_{EN_{s, e}}) = \left(M_{s, e} \vee \left(\bigvee_{s' < s} EN_{s', e} \right) \right) \wedge Con(s)$$

^۶Covering

که در آن ها داریم:

$$Con(s) = \left(\bigwedge_{1 \leq j < j' \leq n \wedge e_j, e_{j'} \in s} \neg C_{e_j, e_{j'}} \right)$$

$$Min(s, e) = \left(\bigwedge_{s' \subseteq E, (s' \subset s \vee s \subset s') \wedge e \notin s'} \neg M_{s', e} \right)$$

فرض کنید که \mathbb{E} مجموعه ای تمامی سه تایی ها به فرم $(E, \#, \vdash')$ باشد که داشته باشیم:

$$\# \subseteq E \times E$$

$$\vdash' \subseteq \mathcal{P} \times E$$

یک تابع به فرم $ES : \times_{V \in \mathcal{V} \setminus \{PV\}} \mathcal{R}(V) \rightarrow \mathbb{E}$ تعریف می کنیم که به صورت شهودی ساختمان رویداد حاصل از مقدار فعلی متغیرها در مدل علی را به دست می دهد. فرض کنیم \vec{v} برداری شامل مقادیر متغیرهای $\mathcal{V} \setminus \{PV\}$ باشد. به ازای هر متغیر مانند $V \in \mathcal{V}$ مقدار آن در \vec{v} را با $\vec{v}(V)$ نمایش می دهیم. تابع ES را به گونه ای تعریف می کنیم که اگر $ES(\vec{v}) = (E, \#, \vdash')$ آنگاه داشته باشیم:

$$\forall e, e' \in E. e \# e' \wedge e' \# e \iff \vec{v}(C_{e, e'}) = T$$

$$\forall s \in \mathcal{P}(E), e \in E. s \vdash' e \iff \vec{v}(EN_{s, e}) = T$$

رفتار ناامن^۷ سیستمی که در قالب ساختمان رویداد مدل شده است، در قالب تابع متغیر PV توصیف شده است و در صورتی که رفتار ناامن در سیستم وجود داشته باشد مقدار آن درست و در غیر این صورت غلط است. در نهایت مقداردهی های مجاز \mathcal{E} را مجموعه ای مقداردهی هایی مانند \vec{v} در نظر می گیریم که خروجی تابع ES به ازای آن ها یک ساختمان رویداد باشد.

برای پیدا کردن علت واقعی در این مدل از تعریف زیر که در واقع سازگار شده ی تعریف برای مدل علی تعمیم یافته است استفاده می کنیم:

تعریف ۱.۳.۲. اگر $M = (\mathcal{S}, \mathcal{F}, \mathcal{E})$ یک مدل علی تعمیم یافته و \vec{V} برداری از متغیرهای $\mathcal{V} \setminus PV$ باشد،

⁷Unsafe Behavior

فرمول $\vec{X} = \vec{x}$ علت واقعی φ در (M, \vec{u}) است اگر شرایط زیر برای آن برقرار باشد:

$$1. (M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \varphi$$

۲. یک افزاز مانند (\vec{Z}, \vec{W}) از مجموعه‌ی متغیرهای \mathcal{V} با شرط $\vec{X} \subseteq \vec{Z}$ و مقادیر (\vec{x}, \vec{w}') برای متغیرهای (\vec{X}, \vec{W}) وجود داشته باشد که داشته باشیم $\vec{Z} = \vec{z}^*$ و $(M, \vec{u}) \models \vec{Z}$ و شرایط زیر را برآورده کند:

$$(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}'] \neg \varphi \wedge \vec{V} = \vec{v} \wedge \vec{v} \in \mathcal{E} \quad (\text{I})$$

$$(\text{ب}) \quad \forall \vec{W}' \subseteq \vec{W}, \vec{Z}' \in \vec{Z}. (M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*] \vec{V} = \vec{v} \wedge \vec{v} \in \mathcal{E} \Rightarrow \varphi$$

۳. \vec{X} مینیمال باشد.

در این تعریف برای اینکه تنها مقداردهی‌های مجاز برای پیدا کردن علت واقعی در نظر گرفته شوند شرط ۲ تغییر است. شرط اضافه شده به شرط ۲. الف بیان می‌کند که تغییرات ایجاد شده در مدل یک مقداردهی مجاز باشند. شرط اضافه شده در ۲. ب اضافه شده است تا این شرط تنها در حالت‌هایی بررسی شود که مقداردهی داده شده مجاز باشد.

۴.۲ پیدا کردن علت خطا در نت‌کت پویا

با استفاده از تعاریف بخش‌های قبلی در این بخش به چگونگی پیدا کردن علت خطا در یک برنامه توصیف شده در نت‌کت پویا می‌پردازیم.

فرض می‌کنیم که یک عبارت نت‌کت پویا p در اختیار داریم. ابتدا عبارت p را به فرم نرمال در می‌آوریم. فرض کنیم عبارت معادل p در فرم نرمال عبارت q باشد. اکنون فرض کنیم $E = \llbracket q \rrbracket$ ساختمان رویداد معادل q باشد. اکنون مدل علی \mathcal{M} را بر اساس E می‌سازیم و رفتار ناامن را در قالب تابع متغیر PV این مدل و به شکل یک شرط بر روی مجموعه‌ی پیکربندی‌های مدل علی توصیف می‌کنیم. در نهایت کافی است برای پیدا کردن علت واقعی رفتار ناامن، علت واقعی $PV = T$ در \mathcal{M} را بر اساس تعریف ۱.۳.۲ پیدا کنیم.

مراجع

- [1] Anderson, Carolyn Jane, Foster, Nate, Guha, Arjun, Jeannin, Jean-Baptiste, Kozen, Dexter, Schlesinger, Cole, and Walker, David. Netkat: Semantic foundations for networks. *Acm sigplan notices*, 49(1):113–126, 2014. [4](#)
- [2] Caltais, Georgiana, Hojjat, Hossein, Mousavi, Mohammad, and Tunc, Hunkar Can. Dynetkat: An algebra of dynamic networks. *arXiv preprint arXiv:2102.10035*, 2021. [11](#), [30](#)
- [3] Foerster, Klaus-Tycho, Schmid, Stefan, and Vissicchio, Stefano. Survey of consistent software-defined network updates. *IEEE Communications Surveys & Tutorials*, 21(2):1435–1461, 2018. [4](#)
- [4] Halpern, Joseph Y. and Pearl, Judea. Causes and explanations: A structural-model approach, part i: Causes. *arXiv:cs/0011012*, Nov 2005. arXiv: cs/0011012. [19](#), [21](#), [24](#)
- [5] Kozen, Dexter. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19(3):427–443, 1997. [4](#)
- [6] McKeown, Nick, Anderson, Tom, Balakrishnan, Hari, Parulkar, Guru, Peterson, Larry, Rexford, Jennifer, Shenker, Scott, and Turner, Jonathan. Openflow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2):69–74, 2008. [4](#)
- [7] Winskel, Glynn. *Event structures*, volume 255 of *Lecture Notes in Computer Science*, page 325–392. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987. [16](#), [25](#)

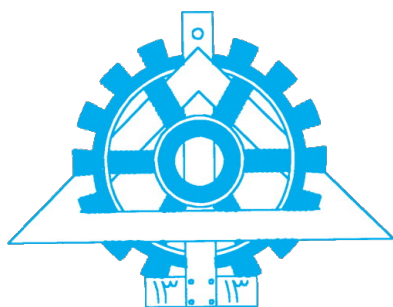
واژه‌نامهٔ فارسی به انگلیسی

واژه‌نامه انگلیسی به فارسی

Abstract

This thesis studies on writing projects, theses and dissertations using tehran-thesis class. It ...

Keywords Writing Thesis, Template, L^AT_EX, X_YY Persian



University of Tehran
College of Engineering
Faculty of Electrical and
Computer Engineering



Explaining Failures in Software-Defined Networks Using Casual Reasoning

A Thesis submitted to the Graduate Studies Office
In partial fulfillment of the requirements for
The degree of Master of Science
in Computer Engineering - Software Engineering

By:

Amir Hossein Seyhani

Supervisors:

Dr. Hossein Hojjat and Dr. Mohammad Reza Mousavi

Advisor:

First Advisor

September 2022