



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: مقدمه
۳	۱.۱ موضوع پژوهش
۴	۲.۱ اهداف پژوهش
۵	۳.۱ ساختار فصل‌ها
۷	فصل ۲: تعاریف و دانش پیش‌زمینه
۷	۱.۲ مقدمه
۷	۲.۲ شبکه‌های مبتنی بر نرم‌افزار
۸	۳.۲ نت‌کت
۹	۱.۳.۲ دستور زبان نت‌کت
۹	۲.۳.۲ معنای نت‌کت
۱۱	۳.۳.۲ توصیف رفتار شبکه با نت‌کت
۱۴	۴.۳.۲ درستی سنجی برنامه‌های نت‌کت
۱۵	۴.۲ نت‌کت پویا
۱۵	۱.۴.۲ دستور زبان نت‌کت پویا
۱۶	۲.۴.۲ معنای عملیاتی نت‌کت پویا
۱۸	۳.۴.۲ توصیف برنامه‌ها در نت‌کت پویا
۲۰	۵.۲ ساختمان رویداد
۲۲	۶.۲ مدل علی
۲۵	۱.۶.۲ علت واقعی

۲۷	۲.۶.۲	پیدا کردن علت واقعی در مسائل
۲۸	۳.۶.۲	مدل تعمیم‌یافته
۲۹	۴.۶.۲	علت واقعی بدون شاهد
۳۱		فصل ۳: روش و راه‌حل پیشنهادی
۳۱	۱.۳	مقدمه
۳۱	۲.۳	معنای عبارات نت‌کت پویا در قالب ساختمان رویداد
۳۳	۱.۲.۳	ترکیب ساختمان رویدادهای برچسب‌دار
۳۵	۲.۲.۳	معنای عبارات نت‌کت پویا در قالب ساختمان رویداد برچسب‌دار
۳۶	۳.۳	مدل علی برای ساختمان رویداد
۳۹	۴.۳	پیدا کردن علت خطا در نت‌کت پویا
۴۱		فصل ۴: نتایج
۴۱	۱.۴	مقدمه
۴۱	۲.۴	لیست سیاه
۴۵	۳.۴	بدون دور بودن
۴۸	۴.۴	نبود سیاه‌چاله
۵۳		فصل ۵: جمع‌بندی و کارهای آینده
۵۳	۱.۵	جمع‌بندی کارهای انجام‌شده
۵۴	۲.۵	نوآوری‌ها و دستاوردها
۵۴	۳.۵	محدودیت‌ها
۵۴	۱.۳.۵	پیچیدگی زمانی
۵۵	۲.۳.۵	توصیف خطا در سطح مدل علی
۵۵	۴.۵	کارهای آینده
۵۵	۱.۴.۵	ترکیب علت
۵۵	۲.۴.۵	سنتز تعمیر

۵۷	فصل ۶: مروری بر کارهای پیشین
۵۷	۱.۶ تخمین پوشش
۵۸	۲.۶ علت خطا در مثال نقض
۵۹	۳.۶ چک کردن علیت
۶۰	۴.۶ علت واقعی در خودکاره‌های زمان‌دار
۶۰	۵.۶ چارچوب علیت بر اساس رد سیستم
۶۰	۶.۶ استدلال مبتنی بر علیت در HML

۶۳	مراجع
----	-------

سوم	واژه‌نامه فارسی به انگلیسی
-----	----------------------------

پنجم	واژه‌نامه انگلیسی به فارسی
------	----------------------------

فهرست کارهای باقیمانده

فصل ۱

مقدمه

در این فصل به صورت اجمالی به موضوع و اهداف مورد انتظار این پژوهش پرداخته می‌شود و ساختار و محتوای فصول این نوشتار توضیح داده می‌شود.

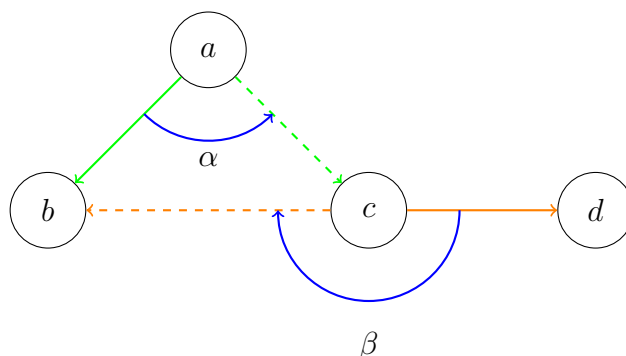
۱.۱ موضوع پژوهش

شبکه‌های کامپیوتری یکی از مهم‌ترین اجزای زیرساخت سیستم‌های کامپیوتری هستند [۸]. شبکه‌های مبتنی بر نرم‌افزار با متمرکز کردن رفتار کنترل‌کننده‌ی شبکه و ساده‌تر کردن عناصر شبکه در سطح داده^۱ تست و درستی‌سنجی شبکه‌ها را تسهیل کرده‌اند. با این وجود به دلیل اینکه هر شبکه‌ی کامپیوتری ذاتاً یک سیستم توزیع‌شده و ناهمگام^۲ است اطمینان از درستی شبکه‌ها حتی با وجود شبکه‌های مبتنی بر نرم‌افزار فرآیندی پیچیده و سخت است. اما این همه‌ی ماجرا نیست. درستی‌سنجی و تست یک شبکه‌ی مبتنی بر نرم‌افزار یا به طور کلی یک نرم‌افزار کامپیوتری قدم اول در فرآیند اشکال‌زدایی^۳ آن است. روش‌های درستی‌سنجی نرم‌افزار در صورتی که سیستم مورد آزمون ویژگی مورد نظر را برآورده نکند یک مثال نقض پیدا کرده و کاربر بر می‌گردانند. به جز این مثال نقض اطلاعات دیگری به کاربر داده نمی‌شود و در نتیجه برای رفع مشکل سیستم کاربر مجبور است تا با روش‌های ابتکاری و با استفاده از دانش و شهود خود در مورد سیستم منشا مشکل را پیدا، آن را برطرف و

^۱Data Plane

^۲Asynchronous

^۳Debug



شکل ۱.۱

فرآیند درستی‌سنجی را تکرار کند. انگیزه اصلی این پژوهش جایگزین کردن بخش انسانی این فرآیند با روش‌های خودکار است. در واقع در این پژوهش به دنبال پیدا کردن علت واقعی رخ دادن خطا در یک شبکه‌ی مبتنی بر نرم‌افزار هستیم.

به عنوان مثال شبکه‌ی رسم‌شده در شکل ۱.۱ را در نظر بگیرید. در این شبکه امکان انجام دو به روز رسانی α و β وجود دارد که به ترتیب مسیرهای سبز و نارنجی پررنگ را با مسیرهای خط‌چین جایگزین می‌کنند. فرض کنید که ویژگی مورد انتظار در این شبکه نرسیدن بسته‌ای از a به d باشد. پس از درستی‌سنجی می‌توان به مثال نقضی برای این ویژگی دست‌یافت که در آن ابتدا به روزرسانی α انجام می‌شود و سپس یک بسته از a به d ارسال می‌شود. این مثال نقض ساده و کوچک است ولی اطلاعاتی در مورد ریشه یا علت به وجود آمدن این خطا به دست نمی‌دهد. با بررسی بیشتر این شبکه می‌توان دریافت که انجام شدن به روز رسانی α پیش از تکمیل شدن به روز رسانی β سبب به وجود آمدن مسیر بین a و d و در نتیجه بروز خطا می‌شود. هدف از انجام این پژوهش پیدا کردن چنین عواملی به عنوان علت واقعی رخ دادن خطا است.

۲.۱ اهداف پژوهش

شبکه‌های مبتنی بر نرم‌افزار به دلیل اینکه ذاتاً توزیع‌شده و ناهمگام هستند مورد استفاده‌ی خوبی برای پیدا کردن علت واقعی خطا و تسهیل فرآیند رفع اشکال هستند. به همین دلیل در این پژوهش سعی شد در این دامنه از مسائل مورد بررسی قرار گرفت تا کارایی آن ارزیابی شود. برای توصیف این شبکه‌ها از زبان نتکت پویا استفاده شده است که یک زبان مینیمال و ساده بر پایه‌ی زبان نتکت است که امکان انجام به روزرسانی را در شبکه فراهم

می‌کند. هدف اصلی این پژوهش ارائه‌ی یک فرمولاسیون از علت واقعی بر اساس تعریف ارائه شده در [۱۲] برای برنامه‌های توصیف شده در زبان نتکت پویا و بررسی کارایی علت‌های واقعی پیدا شده در فرآیند اشکال‌زدایی از شبکه است.

۳.۱ ساختار فصل‌ها

در فصل دوم تعاریف و دانش پیش‌زمینه‌ی مورد نیاز برای بقیه فصول بیان می‌شود. فصل سوم روش ترجمه‌ی یک برنامه‌ی توصیف شده در زبان نتکت پویا به یک مدل علی بیان می‌شود. فصل چهارم شامل به کارگیری روش ارائه شده در این پژوهش برای پیدا کردن علت خطا در چند دسته از ویژگی‌های رایج در شبکه است. در این فصل بررسی می‌شود که علت واقعی پیدا شده تا چه میزان با شهود موجود از مساله تطابق دارد و این فرمولاسیون تا چه حد موفق عمل می‌کند. فصل پنج شامل جمع‌بندی کارهای انجام شده در این پژوهش و بحث در مورد کاستی‌های آن و کارهای پیش‌رو است. در نهایت در فصل ششم مروری بر کارهای پیشین و مرتبط با این پژوهش انجام شده است.

فصل ۲

تعاریف و دانش پیش زمینه

۱.۲ 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.

۲.۲ 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 [۸]. شبکه‌های مبتنی بر نرم‌افزار^۵ یک پارادایم جدید برای طراحی و پیاده‌سازی شبکه‌های کامپیوتری هستند که علاوه بر اینکه مدیریت و درستی سنجی 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 [۱۹] 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 درستی سنجی در مورد رفتار شبکه آسان‌تر شود.

۳.۲ 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 [۱۶] 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

۱.۳.۲ 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:

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

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

۲.۳.۲ 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:

$$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad (۱.۲)$$

$$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad (۲.۲)$$

$$dup \cdot f = n \equiv f = n \cdot dup \quad (۳.۲)$$

$$f \leftarrow n \cdot f = n \equiv f \leftarrow n \quad (۴.۲)$$

$$f = n \cdot f \leftarrow n \equiv f = n \quad (۵.۲)$$

$$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n' \quad (۶.۲)$$

$$f = n \cdot f = n' \equiv 0, \text{ if } n \neq n' \quad (۷.۲)$$

$$\sum_i f = i \equiv 1 \quad (۸.۲)$$

اصل‌های ۱.۲، ۲.۲، ۳.۲ خواص جابه‌جایی ^{۱۹} عملیات‌ها را بیان می‌کنند. اصل ۴.۲ 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 ۵.۲ مشخص شده. اصل ۶.۲ بیان می‌کند که در دنباله‌ای از اختصاص مقادیر به یک فیلد مشخص، تنها آخرین اختصاص تاثیر دارد. در اصل ۷.۲ مشخص شده است که مقدار یک فیلد نمی‌تواند دو مقدار متفاوت داشته باشد. در نهایت اصل ۸.۲ بیان می‌کند که عملیات مجموع فیلترها به ازای هر مقدار ممکن برای یک فیلد مشخص برابر عنصر همانی ^{۲۱} است.

۳.۳.۲ توصیف رفتار شبکه با نت‌کت

در ادامه نحوه‌ی توصیف یک شبکه با استفاده از نت‌کت بیان می‌شود. در شکل ۱.۲ شبکه شامل دو سویچ

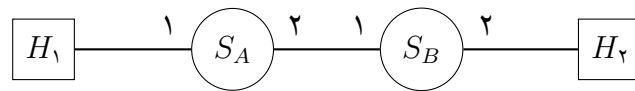
¹⁷ Axiom

¹⁸ Sound and Complete

¹⁹ Commutative

²⁰ Assignment

²¹ Identity



شکل ۱.۲: 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 جابه جایی همه‌ی بسته‌ها به غیر از بسته‌هایی که از نوع SSH هستند وجود داشته باشد. عبارت نت کت 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 می فرستد. این سیاست ^{۲۴} به سادگی رفتار سویچ‌ها را در نت کت تعریف می کند. در ادامه می توان 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 ۱.۲ کافی نیست. برای تکمیل این عبارت لازم است تا رفتار توپولوژی ^{۲۶} شبکه هم به آن افزوده شود. در نت کت توپولوژی شبکه به عنوان یک گراف جهت دار در نظر گرفته می شود و رفتار آن در قالب اجتماع رفتار هر یک از پیوندهای ^{۲۷} آن توصیف می شود. برای شبکه‌ی

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

شکل ۱.۲ 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 ۱.۲ 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 ۱.۲ 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 آنها 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.

۴.۳.۲ 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 ۱.۲ 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 ۱.۲ بسته‌های غیر SSH از هاست ۱ را عبور می دهد کافی است تا درستی نامعادله‌ی زیر را بررسی کرد:

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

۴.۲ نتکت پویا

نتکت پویا^{۲۹} برای رفع برخی از کاستی‌های نتکت ارائه شده است [۴]. به صورت دقیق‌تر نتکت پویا، امکان توصیف به‌روزرسانی سیاست‌های شبکه و همچنین رفتار شبکه در مقابل چندین بسته را ممکن می‌سازد.

۱.۴.۲ 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

۲.۴.۲ معنای عملیاتی نکت پویا

معنای عملیاتی^{۳۳} نکت پویا با استفاده از عبارت‌هایی به شکل (d, H, H') تعریف می‌شوند که d عبارت نکت 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 $e :: l$ 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 Semantic^{۳۴}Synchronized

زیر معنای عملیاتی نتکت پویا را تعریف می‌کنند:

$$(cpol_{\neg}^{\vee}) \frac{\sigma' \in \llbracket p \rrbracket (\sigma :: \langle \rangle)}{(p; q, \sigma :: H, H') \xrightarrow{(\sigma, \sigma')} (q, H, \sigma' :: H')} \quad (9.2)$$

$$(cpol_X) \frac{(p, H_0, H_1) \xrightarrow{\gamma} (p', H'_0, H'_1)}{(X, H_0, H_1) \xrightarrow{\gamma} (p', H'_0, H'_1)} X \triangleq p \quad (10.2)$$

$$(cpol_{\oplus}) \frac{(p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)}{(p \oplus q, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)} \quad (11.2)$$

$$(cpol_{\oplus-}) \frac{(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}{(p \oplus q, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)} \quad (12.2)$$

$$(cpol_{\parallel}) \frac{(p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)}{(p \parallel q, H_0, H'_0) \xrightarrow{\gamma} (p' \parallel q, H_1, H'_1)} \quad (13.2)$$

$$(cpol_{\parallel-}) \frac{(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}{(p \parallel q, H_0, H'_0) \xrightarrow{\gamma} (p \parallel q', H_1, H'_1)} \quad (14.2)$$

$$(cpol_{?}) \frac{}{(x?p; q, H, H') \xrightarrow{x?p} (q, H, H')} \quad (15.2)$$

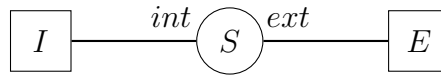
$$(cpol_{!}) \frac{}{(x!p; q, H, H') \xrightarrow{x!p} (q, H, H')} \quad (16.2)$$

$$(cpol_{!?}) \frac{(q, H, H') \xrightarrow{x!p} (q', H, H') (s, H, H') \xrightarrow{x?p} (s', H, H')}{(q \parallel, H, H') \xrightarrow{rcfg(x,p)} (q' \parallel s', H, H')} \quad (17.2)$$

$$(cpol_{?!}) \frac{(q, H, H') \xrightarrow{x?p} (q', H, H') (s, H, H') \xrightarrow{x!p} (s', H, H')}{(q \parallel, H, H') \xrightarrow{rcfg(x,p)} (q' \parallel s', H, H')} \quad (18.2)$$

قانون ۹.۲ 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 ۱۰.۲ بیان می‌کند که رفتار متغیر X که برابر با عبارت p است معادل با رفتار عبارت p است. قوانین ۱۱.۲ و ۱۲.۲ رفتار غیرقطعی را توصیف می‌کنند. قوانین ۱۳.۲ و ۱۴.۲ رفتار دو عبارت موازی را توصیف می‌کنند. قوانین ۱۵.۲ و ۱۶.۲ 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 ۱۵.۲ و ۱۶.۲ توصیف شده است.

³⁵Synchronization



شکل ۲.۲: مثال دیوار آتش

۳.۴.۲ 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 ۲.۲ 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:

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

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

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

$$secConReq?1; Switch'$$

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

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

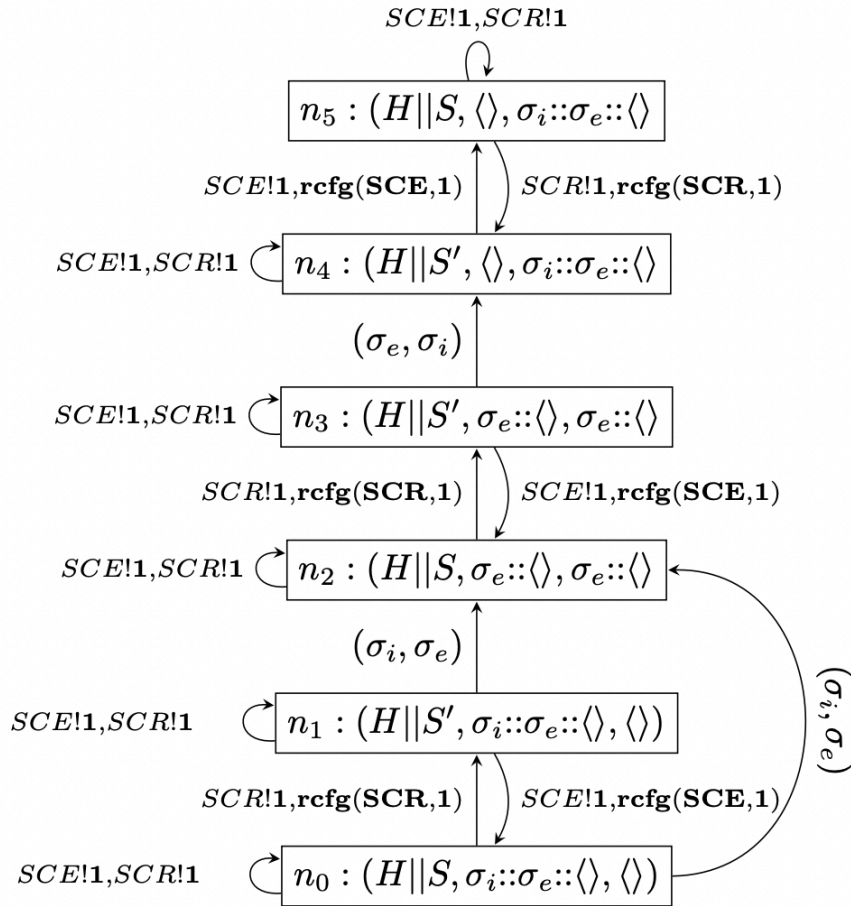
$$secConEnd?1; Switch$$

$$Init \triangleq Host \parallel Switch$$

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

^{۳۶}Firewall^{۳۷}Stateful

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 ۳.۲ سیستم انتقال برچسب‌دار^{۳۸} این شبکه را در حالتی



شکل ۳.۲: 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

۵.۲ ساختمان رویداد

ساختمان رویداد ^{۳۹} [۲۱] یک مدل محاسباتی ^{۴۰} غیرجای گذاری شده ^{۴۱} برای پردازش‌های همروند ^{۴۲} است. در این مدل، برخلاف مدل‌های جایگذاری شده ^{۴۳} مانند سیستم انتقال که همروندی پردازش‌های موازی با انتخاب غیرقطعی مدل می‌شود، همروندی پردازش‌ها به صورت صریح در مدل توصیف می‌شوند.

تعریف ۱.۵.۲. ساختمان رویداد یک ساختمان رویداد یک سه تایی $(E, \#, \vdash)$ 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 فعال سازی ^{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 ۲.۵.۲. 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 ۳.۵.۲. 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:

$$1. \quad x \in Con$$

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

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

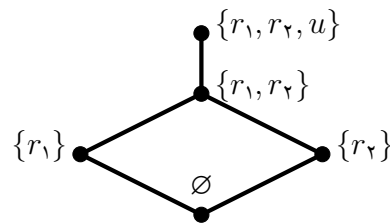
شبکه‌ی موجود در شکل ۵.۲ را در نظر بگیرید. در این شبکه دو هاست ۱ و ۲ به صورت هم‌رند یک بسته را به سویچ ارسال می‌کنند. این بسته‌ها شامل اطلاعات برای به روزرسانی مسیرهای دیگر در شبکه هستند، بنابراین سویچ پس از دریافت هر دوی این بسته‌ها آن‌ها را پردازش کرده و مسیرهای خود را به روزرسانی می‌کند. برای مدل کردن این شبکه می‌توانیم از یک ساختمان رویداد به صورت زیر استفاده کنیم:

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

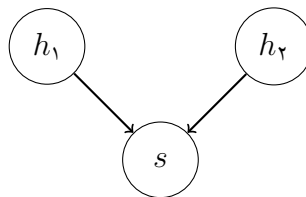
در این ساختمان رویداد، رویدادها به ترتیب دریافت یک بسته از هاست ۱، دریافت یک بسته از هاست ۲ و به روزرسانی سویچ را مدل می‌کنند. یکی از روش‌های رسم نمودار برای ساختمان رویداد، رسم نمودار هس^{۴۷} برای مجموعه‌ی پیکربندی‌های این ساختمان رویداد بر اساس رابطه‌ی زیرمجموعه است. برای مثالی که بیان شد می‌توان نموداری مطابق شکل ۴.۲ را رسم کرد.

⁴⁶ Configuration

⁴⁷ Hasse



شکل ۴.۲



شکل ۵.۲

۶.۲ 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 [۱۲] 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 $(X = u + 4) \leftarrow [Y \leftarrow 4]$ (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 ۱.۶.۲. 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

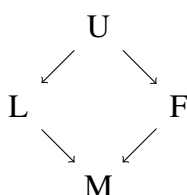
- متغیر L که اگر رعد و برق اتفاق افتاده باشد مقدار آن درست است و در غیر این صورت غلط است

- 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 ۱.۶.۲ 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 [۱۲] 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 ۱.۶.۲ را نشان می‌دهد:



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

۱.۶.۲ 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 $\psi(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}] (X = x)$ به عنوان مثال $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}] (X = x)$ نشان می‌دهیم. به عنوان مثال اگر مقدار متغیر X در راه حل معادلات مدل $M_{\vec{Y} \leftarrow \vec{y}}$ تحت هم‌بافت \vec{u} برابر x باشد.

تعریف ۲.۶.۲. 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:

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

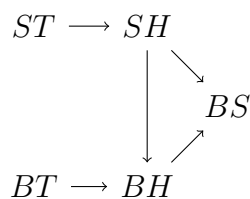
$$(ب) \quad \forall \vec{W}' \subseteq \vec{W}, \vec{Z}' \in \vec{Z}. (M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*] \varphi$$

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 ۶.۲

۲.۶.۲ 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 ۶.۲ 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 ۲.۲ الف برقرار است. اما به ازای $\bar{Z}' = \{BH\}$ شرط ۲.۲ 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.

۳.۶.۲ 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 [۱۲] 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

۴.۶.۲ علت واقعی بدون شاهد

فرض کنید که $\vec{X} = \vec{x}$ یک علت واقعی برای φ در (M, \vec{u}) با استفاده از شاهد $(\emptyset, \emptyset, \emptyset)$ باشد. با توجه به اینکه در اینجا \vec{W} یک بردار خالی است پس عملاً شرط ۲.ب به بررسی شرط زیر تبدیل می‌شود:

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

با دقت در شرط بالا می‌توان دریافت که مقدار متغیرها در فرمول‌های $[\vec{X} \leftarrow \vec{x}, \vec{Z}' \leftarrow \vec{z}^*] \varphi$ با مقدار متغیرها در هم‌بافت اولیه تفاوتی ندارد زیرا مقدار آن‌ها به مقداری که در هم‌بافت اولیه داشته‌اند برگردانده می‌شود. بنابراین در شرط بالا می‌توان نتیجه گرفت:

$$(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{Z}' \leftarrow \vec{z}^*] \varphi \iff (M, \vec{u}) \models \varphi$$

بنابراین شرط ۲.ب معادل با شرط ۱ می‌شود. با توجه به این موضوع می‌توان قضیه زیر را نتیجه گرفت:

گزاره ۳.۶.۲. اگر $\vec{X} = \vec{x}$ در (M, \vec{u}) بدون شاهد شرط‌های ۱، ۲.الف و ۳ در تعریف ۲.۶.۲ را برای φ برآورده کند آنگاه $\vec{X} = \vec{x}$ یک علت واقعی برای φ در (M, \vec{u}) است.

فصل ۳

روش و راه حل پیشنهادی

۱.۳ 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.

۲.۳ 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 [۲۱] می‌باشد.

تعریف ۱.۲.۳. محدودیت فرض کنید $E = (E, \#, \vdash)$ یک ساختمان رویداد باشد. به ازای یک مجموعه‌ی $A \subseteq E$ محدودیت^۱ E به A یک ساختمان رویداد به صورت زیر است:

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

که داشته باشیم:

$$X \subseteq Con_A \iff X \subseteq A \wedge X \in Con$$

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

تعریف ۲.۲.۳. 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 ۳.۲.۳. 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 ۴.۲.۳. 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

۱.۲.۳ ترکیب ساختمان رویدادهای برچسب‌دار

تعریف ۵.۲.۳. 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 ۶.۲.۳. 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 ۷.۲.۳. 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 $E_i \rightarrow_* E$ 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 ۸.۲.۳. 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.

۲.۲.۳ 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 [۴] 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 ۹.۲.۳. 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 [۴] 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:

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

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

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

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

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

۳.۳ 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 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:

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

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

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

⁶Covering

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 ی 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 ی 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 ۱.۳.۳. 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:

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

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

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.

۴.۳ 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 ۱.۳.۳ پیدا کنیم.

فصل ۴

نتایج

۱.۴ مقدمه

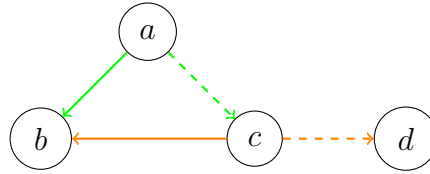
در این فصل با استفاده از مدل علی تعریف شده در فصل پیشین، علت نقض چند رسته از ویژگی‌ها در شبکه را مورد بررسی قرار می‌دهیم. در ادامه فرض می‌کنیم که فیلد sw در همه‌ی توصیف‌های نکت پویا وجود دارد. همچنین برای ساده‌تر شدن توصیف‌ها از اصل زیر استفاده می‌کنیم:

$$x \rightarrow y \triangleq sw = x \cdot sw \leftarrow y$$

۲.۴ لیست سیاه

در این ویژگی، یک لیست سیاه^۱ از مکان‌هایی در شبکه وجود دارد که نباید در شبکه به آن‌ها دسترسی وجود داشته باشد [۲۰]. مهم‌ترین استفاده از لیست سیاه را می‌توان برای اعمال سیاست‌های کنترل دسترسی در نظر گرفت که مثلاً برخی از هاست‌ها که دارای اطلاعات حیاتی هستند در لیست سیاه قرار می‌گیرند تا از بیرون به آن‌ها دسترسی وجود نداشته باشد. به عنوان مثال دیگر ممکن است برخی از عناصر شبکه برای تعمیر برای مدتی

¹Blacklist



شکل ۱.۴

کنار گذاشته شوند برای این منظور می‌توان آن‌ها را در لیست سیاه قرار داد تا دسترسی به آن‌ها سبب از دست رفتن بسته‌ها نشود. برای پیدا کردن علت نقض شدن ویژگی لیست سیاه شبکه‌ی رسم شده در شکل ۱.۴ را در نظر بگیرید. در این شبکه سوئیچ d در لیست سیاه قرار دارد، بنابراین در هیچ لحظه نباید از a که ورودی شبکه است در دسترس باشد. در شبکه‌ی بالا ابتدا مسیریایی که با خط پررنگ مشخص شده‌اند وجود دارند. در ادامه هر یک از مسیرها با مسیرهای خط‌چین جایگزین می‌شوند. فرض کنید به روز رسانی این مسیرها توسط دو پردازنده هم‌روند انجام می‌شود. واضح است که اگر هر دوی این به‌روز رسانی‌ها انجام شوند دسترسی به سوئیچی که در لیست سیاه قرار دارد ممکن می‌شود. اکنون فرض کنید که از عبارات زیر برای توصیف این شبکه در نکت پویا استفاده کنیم:

$$P = p!1$$

$$F_p = a \rightarrow c \oplus c \rightarrow b \oplus a \rightarrow b$$

$$Q = q!1$$

$$F_q = a \rightarrow b \oplus c \rightarrow d$$

$$N = F \oplus p?1; N_p \oplus q?1; N_q$$

$$F_{pq} = a \rightarrow c \oplus c \rightarrow d \oplus a \rightarrow d$$

$$N_p = F_p \oplus q?1; F_{pq}$$

$$SDN = \delta_{\mathcal{L}}(N \parallel P \parallel Q)$$

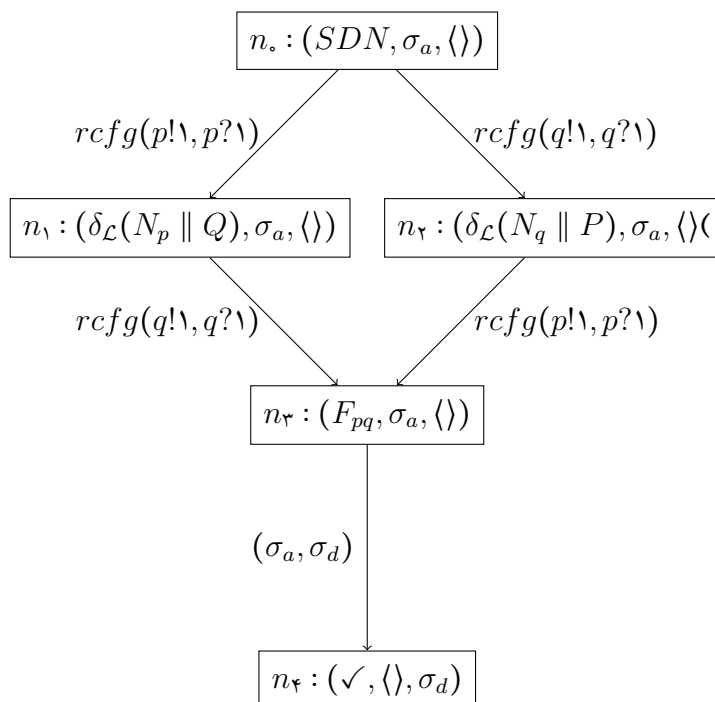
$$N_q = F_q \oplus p?1; F_{pq}$$

$$\mathcal{L} = \{p!1, p?1, q?1, q!1\}$$

$$F = a \rightarrow b \oplus c \rightarrow b$$

در توصیف بالا پردازنده‌های P و Q به ترتیب وظیفه‌ای ارسال پیام برای به روز رسانی مسیرهای سبز و نارنجی را دارند. پردازنده‌ی N رفتار ابتدایی شبکه و پردازنده‌های N_p و N_q به ترتیب رفتار شبکه را پس از به روز رسانی مسیرهای سبز و نارنجی توصیف می‌کنند. پردازنده‌های F, F_p, F_q, F_{pq} رفتارهای ارسالی^۲ شبکه را توصیف می‌کنند. در نهایت رفتار کلی شبکه توسط پردازنده‌ی SDN توصیف شده است که حاصل ترکیب موازی پردازنده‌های N, P, Q

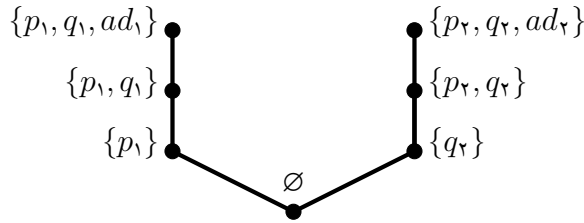
^۲Forwarding



شکل ۲.۴

و جلوگیری از اجرای عملیات‌های همگام نشده است. در توصیف بالا امکان اجرای هر دو به روز رسانی وجود دارد بنابراین شبکه این امکان را دارد که به حالتی برسد که مسیر ارسال ad در آن وجود داشته باشد. برای مثال فرض کنید که σ_a یک بسته وارد شده به شبکه باشد که داشته باشیم: $\sigma_a(sw) = a$. شکل ۲.۴ بخشی از نمودار سیستم حالت این شبکه را زمانی که این بسته به شبکه وارد شود نشان می‌دهد. اگر فرض کنیم σ_d بسته‌ای باشد که $\sigma_d(sw) = d$ همانطور که در نمودار مشخص است دو مسیر به حالتی که بسته از سویچ a به d برسد وجود دارد. همانطور که در نمودار مشخص است به دلیل هم‌روندی پدازه‌های P و Q دو ترتیب برای اجرای این به‌روز رسانی‌ها وجود دارد و به همین دلیل دو مسیر منجر به خطا در این شبکه وجود دارد. اکنون می‌خواهیم علت بروز این خطا را پیدا کنیم. فرض کنید $E = \llbracket SDN \rrbracket$ ساختمان رویداد این شبکه و \mathcal{M} مدل علی E بر اساس تعریف باشد. در این مدل تابع متغیر PV را به صورت زیر تعریف می‌کنیم:

$$F_{PV}(\vec{V}_{PV}) = \exists c \in \mathcal{F}(ES(\vec{v})). \exists e \in c. l(e) = a \rightarrow d$$



شکل ۳.۴

تابع بالا رفتار ناامن را به گونه‌ای تعریف می‌کند که مقدار آن در صورتی درست است که پیکر بندی‌ای در ساختمان رویداد وجود داشته باشد که رویدادی با برچسب $a \rightarrow d$ داشته باشد. با توجه به ترتیب اجرای به‌روزرسانی‌ها در شبکه دو رویداد برای هر یک از عملیات‌های $a \rightarrow d$, $rcfg(p!1, p?1)$, $rcfg(q!1, q?1)$ در ساختمان رویداد وجود دارد. فرض کنید برای رویدادهای مرتبط با این عملیات‌ها شش رویداد $p_1, p_2, q_1, q_2, ad_1, ad_2$ وجود داشته باشد که برچسب آن‌ها به صورت زیر باشد:

$$l(p_1) = rcfg(p!1, p?1)$$

$$l(p_2) = rcfg(p!1, p?1)$$

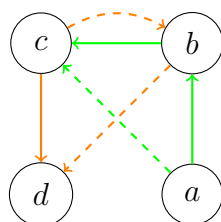
$$l(q_1) = rcfg(q!1, q?1)$$

$$l(q_2) = rcfg(q!1, q?1)$$

$$l(ad_1) = a \rightarrow d$$

$$l(ad_2) = a \rightarrow d$$

شکل ۳.۴ قسمتی از نمودار ساختمان رویداد این شبکه را نشان می‌دهد که در آن تمام پیکر بندی‌هایی که یکی از رویدادهای ad_1 یا ad_2 قابل دسترسی باشد. با استفاده از مدل علی در این مثال می‌توانیم $C(p_1, q_1) = F$ را به عنوان یک علت برای نقض ویژگی لیست سیاه بیان معرفی کنیم در صورتی که از $(C(p_2, q_2), T, T)$ به عنوان شاهد استفاده کنیم. در E پیکر بندی $\{p_1, q_1, ad_1\}$ قابل دسترسی است. بنابراین مقدار PV صحیح است. همچنین بین رویدادهای p_1 و q_1 تعارضی وجود ندارد پس $C(p_1, q_1) = F$. بنابراین شرط ۱ در تعریف برقرار است. اکنون فرض کنید که مقدار $C(p_1, q_1)$ و $C(p_2, q_2)$ را برابر صحیح قرار دهیم. در این حالت هیچ یک از پیکر بندی‌های $\{p_1, q_1, ad_1\}$ و $\{p_2, q_2, ad_2\}$ دیگر نمی‌توانند عضوی از پیکر بندی‌های $ES(\vec{v})$ در \mathcal{M} باشند.



شکل ۴.۴

پس در این حالت مقدار PV غلط می‌شود بنابراین شرط ۲.الف برقرار می‌شود. برای بررسی برقراری شرط ۲.ب باید فرض کنیم که مقدار $C(p_1, q_1)$ غلط است. توجه کنید که در این شرایط پیکربندی $\{p_1, q_1, ad_1\}$ عضوی از پیکربندی‌های $ES(\vec{v})$ است و مقدار $C(p_2, q_2)$ روی این مساله تاثیری ندارد. همچنین برگرداندن مقادیر بقیه متغیرها به مقدار اولیه آن‌ها باعث حذف $\{p_1, q_1, ad_1\}$ از مجموعه‌ی پیکربندی‌ها نمی‌شود بنابراین شرط ۲.ب هم برقرار است. با توجه به اینکه علت تنها شامل یک جمله است بنابراین شرط مینیمال بودن هم برقرار است. بنابراین در نهایت می‌توان نتیجه گرفت که $C(p_1, q_1)$ یک علت واقعی برای بروز خطا در این شبکه است. به صورت شهودی اتفاق افتادن هر دوی به روزرسانی‌ها علت وقوع خطا در این مثال است که توسط تعریف ارائه شده پیدا شده است.

۳.۴ بدون دور بودن

این ویژگی بیان می‌کند که شبکه نباید هرگز دارای دور باشد. [۸]. وجود دور در شبکه می‌تواند باعث مشکلاتی مانند دور زدن یک بسته در شبکه بدون رسیدن به مقصد و در نتیجه کاهش کارایی شبکه شود. به عنوان مثال شبکه‌ی رسم شده در شکل ۴.۴ را در نظر بگیرید. در ابتدا مسیری از a به d وجود دارد. در این شبکه دو به روزرسانی بر روی سویچ‌های a و c انجام می‌شود تا مسیر جدیدی از a به d ایجاد شود که اینبار ابتدا از c

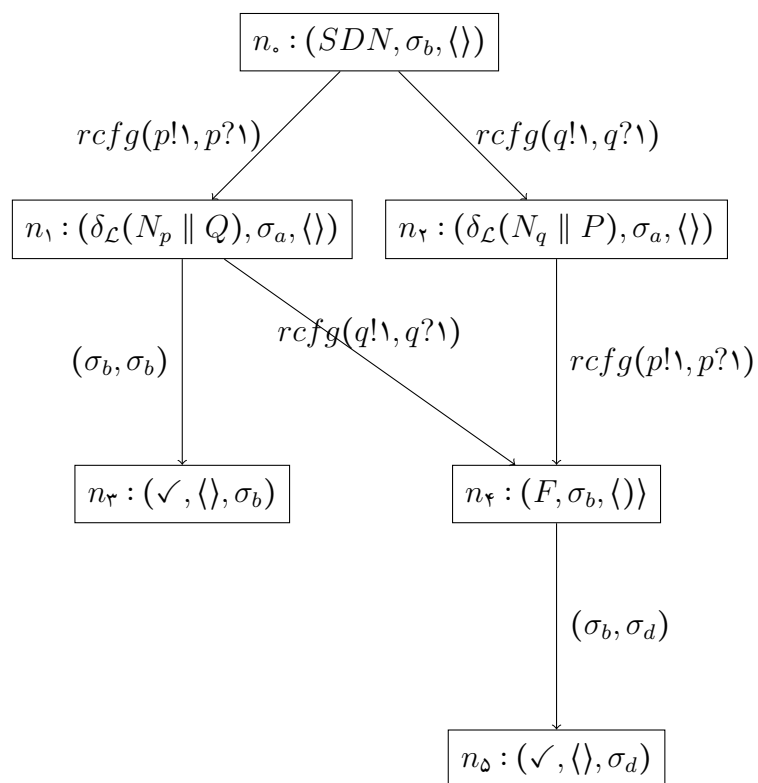
عبور می‌کند. می‌توانیم از توصیف نتکت پویای زیر برای توصیف این شبکه استفاده کنیم:

$$\begin{aligned}
 P &= p!1 & F &= a \rightarrow b \oplus a \rightarrow c \oplus a \rightarrow d \\
 Q &= q!1 & & \oplus b \rightarrow c \oplus b \rightarrow d \oplus c \rightarrow d \\
 N &= F \oplus p?1; N_p \oplus q?1; N_q & F_p &= a \rightarrow c \oplus a \rightarrow d \oplus c \rightarrow d \\
 N_p &= F_p \oplus q?1; F & F_q &= a \rightarrow b \oplus a \rightarrow c \oplus a \rightarrow d \\
 N_q &= F_q \oplus p?1; F & & \oplus b \rightarrow c \oplus b \rightarrow b \oplus b \rightarrow d \\
 SDN &= \delta_{\mathcal{L}}(N \parallel P \parallel Q) & & \oplus c \rightarrow b \oplus c \rightarrow c \oplus c \rightarrow d \\
 \mathcal{L} &= \{p!1, p?1, q!1, q?1\}
 \end{aligned}$$

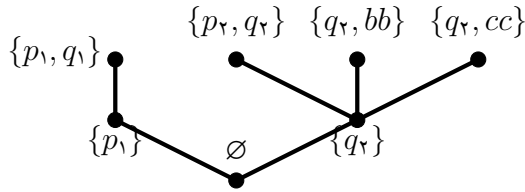
در توصیف بالا پردازش‌های P و Q به ترتیب وظیفه‌ای ارسال پیام برای به روز رسانی مسیرهای سبز و نارنجی را دارند. توجه کنید که در این توصیف پس از اجرای هر دو به روز رسانی رفتار ارسالی شبکه همانند رفتار اولیه خود می‌شود. همانطور که در شکل ۴.۴ مشخص است اگر به روز رسانی نارنجی پیش از به روز رسانی سبز انجام شود در شبکه یک دور شامل گره‌های c و b ایجاد می‌شود. شکل ۵.۴ قسمتی از سیستم انتقال برچسب‌دار شبکه را در حالتی که یک بسته ورودی روی سویچ b وجود داشته باشد را نشان می‌دهد. همانطور که در شکل مشخص است پس از انجام به روز رسانی مسیر نارنجی امکان عملیاتی به شکل (σ_b, σ_b) وجود دارد که به معنی وجود حلقه در این شبکه است. اما اگر به روز رسانی مسیر سبز هم انجام شود تنها عملیات ممکن روی بسته ارسال آن به سویچ d است. اکنون فرض کنید که $E = \llbracket SDN \rrbracket$ ساختمان رویداد این شبکه و \mathcal{M} مدل علی E بر اساس تعریف باشد. در این مدل تابع متغیر PV را به صورت زیر تعریف می‌کنیم:

$$\begin{aligned}
 F_{PV}(\vec{V}_{PV}) &= \bigvee_{c \in C} c \in \mathcal{F}(ES(\vec{v})) \\
 C &= \{c \subset E \mid \exists e \in c. l(e) = b \rightarrow b \vee l(e) = c \rightarrow c\}
 \end{aligned}$$

در این تابع رفتار ناامن وجود پیکربندی ای شامل یکی از برچسب‌های $b \rightarrow b$ یا $c \rightarrow c$ در شبکه است. همانند مثال قبل با توجه به ترتیب اجرای به روز رسانی‌ها در شبکه دور رویداد برای هر یک از عملیات‌های $rcfg(p!1, p?1), rcfg(q!1, q?1)$ در ساختمان رویداد وجود دارد. فرض کنید برای رویدادهای مرتبط با این عملیات‌ها چهار رویداد p_1, p_2, q_1, q_2



شکل ۵.۴



شکل ۶.۴

وجود داشته باشد که برچسب آن‌ها به صورت زیر باشد:

$$l(p_1) = rcfg(p!1, p?1)$$

$$l(p_2) = rcfg(p!1, p?1)$$

$$l(q_1) = rcfg(q!1, q?1)$$

$$l(q_2) = rcfg(q!1, q?1)$$

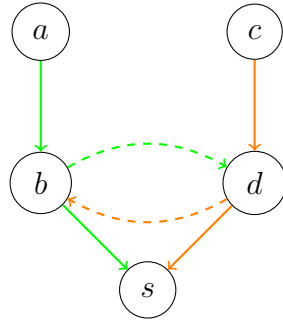
همچنین فرض کنید برچسب رویدادهای bb, cc به ترتیب $b \rightarrow b, c \rightarrow c$ باشد.

شکل ۶.۴ قسمتی از نمودار ساختمان رویداد این شبکه را نشان می‌دهد که در آن تمام پیکربندی‌هایی که یکی از رویدادهای bb یا cc قابل دسترس باشد. در اینجا می‌توان $M(\{p_2\}, q_2) = F$ را به عنوان علت واقعی بروز خطا معرفی کرد. اگر مقدار $M(\{p_2\}, q_2)$ را برابر صحیح قرار دهیم آن‌گاه با توجه به تابع $M(\emptyset, q_2)$ که در بخش ۳.۳ تعریف شده است، مقدار $M(\emptyset, q_2)$ غلط شده و در نتیجه مقدار $EN(\emptyset, q_2)$ هم غلط می‌شود. این مساله باعث می‌شود که دیگر هیچ‌کدام از پیکربندی‌های شاخه‌ی سمت راست شکل ۶.۴ دیگر عضو پیکربندی‌های $ES(\vec{v})$ نباشند و در نتیجه مقدار PV غلط می‌شود. با توجه به اینکه در شاهد \vec{W} خالی است می‌توان نتیجه گرفت که $M(\{p_2\}, q_2) = F$ علت واقعی به وجود آمدن دور در این شبکه است.

۴.۴ نبود سیاه‌چاله

در یک شبکه سیاه‌چاله‌ها^۳ عناصری در شبکه هستند که وظیفه ارسال بسته‌ها را دارند (مثلاً سویچ یا روتر) ولی برخی از بسته‌ها را پس از دریافت به جایی ارسال نمی‌کنند و در واقع مانند سیاه‌چاله این بسته‌ها در آن‌ها

³Blackhole



شکل ۷.۴

گم می‌شوند [۲۰]. در یک شبکه که مکان‌های ورودی و خروجی مشخص دارد عدم وجود سیاه‌چاله در شبکه را می‌توان معادل این ویژگی که همه‌ی بسته‌های ورودی به شبکه از آن خارج شوند دانست. به عنوان مثال شبکه‌ی موجود در شکل ۷.۴ را در نظر بگیرید. فرض کنید که در این شبکه a, c ورودی‌های شبکه و s خروجی شبکه باشد. در این شبکه دو به روز رسانی برای جایگزینی ds با db و bs با bd انجام می‌شود. در این شبکه در حالت ابتدایی و پس از انجام یکی از به روز رسانی‌ها ورودی‌ها به خروجی مسیر وجود دارد اما اگر هر دوی این به روز رسانی‌ها انجام شوند دیگر s قابل دسترسی نیست و عملاً بسته‌های ورودی به شبکه به خروجی نمی‌رسند. این شبکه را می‌توانیم به فرم زیر در نتکت پویا توصیف کنیم:

$$P = p!1$$

$$F = a \rightarrow s \oplus c \rightarrow s$$

$$Q = q!1$$

$$F_{pq} = a \rightarrow b \oplus c \rightarrow d$$

$$N = F \oplus p?1; N_p \oplus q?1; N_q$$

$$SDN = \delta_{\mathcal{L}}(N \parallel P \parallel Q)$$

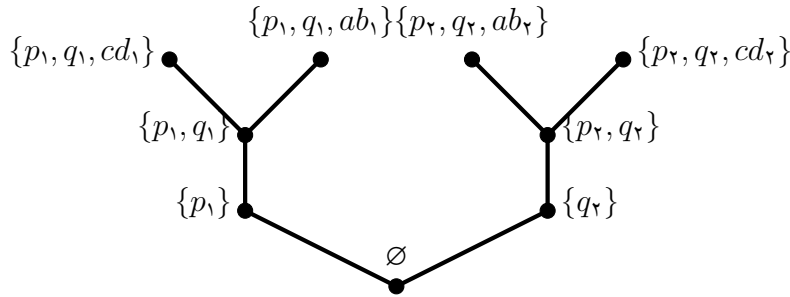
$$N_p = F \oplus q?1; F_{pq}$$

$$\mathcal{L} = \{p!1, p?1, q?1, q!1\}$$

$$N_q = F \oplus p?1; F_{pq}$$

در ادامه فرض کنید که \mathcal{M} مدل علی این شبکه باشد. در این مدل تابع متغیر PV را به صورت زیر تعریف می‌کنیم:

$$F_{PV}(\vec{V}_{PV}) = \exists c \in \mathcal{F}(ES(\vec{v})), \exists e \in c.l(e) = \alpha \cdot \pi \wedge \pi(sw) \neq s$$



شکل ۸.۴

در تعریف این تابع رفتار ناامن وجود پیکربندی ای با برجسب از نوع $\alpha \cdot \pi$ یا به عبارت دیگر وجود رویدادی از نوع ارسال بسته باشد که سوییچ مقصد ارسال آن سوییچ s نباشد. همانند مثال مربوط نبود دور در شبکه، در این مثال هم با توجه به ترتیب اجرای به روز رسانی‌ها دورویداد برای هر یک از عملیات‌های $a \rightarrow rcf g(p!1, p?1), rcf g(q!1, q?1), a \rightarrow b, ac \rightarrow d$ وجود دارد. بنابراین فرض کنید که رویدادهای p_i, q_i, ab_i, cd_i به ازای $i = 1, 2$ در ساختمان رویداد این مدل وجود داشته باشند و برجسب‌گذاری آن‌ها به صورت زیر باشد:

$$l(p_1) = rcf g(p!1, p?1)$$

$$l(p_2) = rcf g(p!1, p?1)$$

$$l(q_1) = rcf g(q!1, q?1)$$

$$l(q_2) = rcf g(q!1, q?1)$$

$$l(ab_1) = a \rightarrow b$$

$$l(ab_2) = a \rightarrow b$$

$$l(cd_1) = c \rightarrow d$$

$$l(cd_2) = c \rightarrow d$$

پیکربندی‌هایی از این ساختمان رویداد که شامل رویدادی با برجسب $a \rightarrow b$ یا $c \rightarrow d$ باشند نقض شدن

ویژگی نبود سیاه‌چاله را نشان می‌دهند. بنابراین تابع متغیر PV را به فرم زیر توصیف می‌کنیم:

$$F_{PV}(\vec{V}_{PV}) = \exists c \in \mathcal{F}(ES(\vec{v})), \exists e \in c.l(e) = \alpha \cdot \pi \wedge \pi(sw) \neq s$$

در این مدل هم همانند مثال نبود دور در شبکه عدم وجود تعارض بین رویدادهای p_1 و q_1 را می‌توان به عنوان علت واقعی نقض شدن ویژگی در نظر گرفت. برای این منظور از شاهد (C_{p_2, q_2}, T, T) استفاده می‌کنیم. واضح است که اگر مقدار هر دو متغیر C_{p_1, q_1} و C_{p_2, q_2} را برابر غلط قرار دهیم آنگاه هیچ یک از پیکربندی‌های $\{p_1, q_1, cd_1\}, \{p_1, q_1, ab_1\}, \{p_2, q_2, ab_2\}, \{p_2, q_2, cd_2\}$ دیگر عضوی از پیکربندی‌های $ES(\vec{v})$ نیستند. از طرفی در شرایطی که C_{p_1, q_1} مقدار درست داشته باشد آنگاه پیکربندی‌های $\{p_1, q_1, ab_1\}, \{p_1, q_1, cd_1\}$ عضو $ES(\vec{v})$ هستند و مقدار متغیر C_{p_2, q_2} تاثیری روی این مساله ندارد بنابراین در نهایت می‌توان نتیجه گرفت که $C_{p_1, q_1} = F$ علت واقعی نقض ویژگی است.

فصل ۵

جمع‌بندی و کارهای آینده

۱.۵ جمع‌بندی کارهای انجام‌شده

در این پژوهش سعی شد تا از مفهوم علت واقعی برای پیدا کردن علت خطا در شبکه‌های نرم‌افزاری استفاده شود. برای این منظور زبان نت‌کت پویا به عنوان زبان برنامه‌ی شبکه انتخاب شد. دلیل انتخاب نت‌کت پویا این است که اولاً این زبان چون بر اساس نت‌کت بنا شده سادگی و مینیمال بودن خود را حفظ کرده است که این مساله به ساده‌تر شدن مساله کمک می‌کند. ثانیاً نت‌کت پویا امکان به روز رسانی شبکه را فراهم می‌کند و از آن‌جایی که هدف نهایی پیدا کردن علت خطا تسهیل کردن فرآیند رفع خطا است، وجود ساختارهایی در زبان برای به روز رسانی‌های شبکه کمک می‌کند تا این ساختارها هم در پیدا کردن علت نقش داشته باشند و کمک بیشتری به رفع خطا کنند. برای اختصاص معنا به عبارات نت‌کت پویا از ساختمان رویداد استفاده شده است. در نهایت برای بر اساس تعریف مدل علی ارائه شده در [۱۲] مدل علی ساختمان رویداد در این پژوهش ارائه شده است. برای بررسی کارایی این مدل چند دسته از ویژگی‌های مطرح شبکه مورد بررسی قرار گرفته‌اند و تطابق علت واقعی پیدا شده با شهود موجود از مساله مورد بحث قرار گرفته است.

۲.۵ نوآوری‌ها و دستاوردها

در این پژوهش سعی شد تا از رویکرد متفاوتی نسبت به روش‌های پیشین برای توصیف خطا استفاده شود. در پژوهش‌هایی مانند [۱۷، ۶، ۲] المان‌هایی در رفتار یا وضعیت سیستم به عنوان علت بروز خطا در نظر گرفته می‌شوند. اما در این پژوهش در میان المان‌های ساختاری سیستم به جستجوی علت پرداخته شده است. یکی از مزایایی که برای این روش می‌توان در نظر گرفت استفاده آسان‌تر از آن برای فرآیند تعمیر یا سنتز خودکار نرم‌افزار است. چون در اینجا ساختارهای سیستم به عنوان علت پیدا می‌شوند این موضوع به تولید تعمیر خودکار کمک می‌کند. مساله‌ی دیگر استفاده از ساختمان رویداد به عنوان مدل معنایی است. همانطور که پیش‌تر بیان شد ساختمان رویداد یک مدل غیرجایگذاری شده است که در آن هم‌روندی به صورت صریح مشخص می‌شود. این موضوع سبب می‌شود که اولاً در فرآیند پیدا کردن علت، ساختارهای جعلی مانند جایگذاری پرده‌های هم‌روند که یک ترتیب برای اجرای آن‌ها ایجاد می‌کند به عنوان علت پیدا نشوند، ثانیاً بتوان هم‌روندی دو پرده یا عملیات را به عنوان علت تعریف کرد، کاری که در مدل‌های جایگذاری شده به دلیل اینکه هم‌روندی صراحتاً توصیف نمی‌شود ممکن نیست. یکی دیگر از تفاوت‌های این روش با روش‌هایی مانند [۱۷، ۶] در استفاده از تعریف علت واقعی است. در پژوهش‌های ذکر شده با اقتباس از تعریف HP تعریف جدیدی برای علت واقعی ارائه شده است و در مورد معادل بودن این تعاریف بحثی نشده است. در مقابل در پژوهش جاری به صورت مستقیم از تعریف HP استفاده شده است.

۳.۵ محدودیت‌ها

۱.۳.۵ پیچیدگی زمانی

یک ساختمان رویداد با n رویداد را در نظر بگیرید. مدل علی این ساختمان رویداد شامل $O(2^n)$ متغیر است. برای پیدا کردن علت واقعی در این مدل و به طور خاص برای بررسی شرط ۲.ب لازم است تا تمامی زیر مجموعه‌های یک افراز از این متغیرها بررسی شود که در بهترین حالت پیچیدگی زمانی $O(2^{2^n})$ دارد. بنابراین پیاده‌سازی این روش بدون بهینه‌سازی یا استفاده از روش‌های ابتکاری عملاً ممکن نیست.

۲.۳.۵ توصیف خطا در سطح مدل علی

در روش ارائه شده در این پژوهش برای پیدا کردن علت خطا در یک برنامه توصیف شده در زبان نت‌کت پویا، لازم است تا رفتار ناامن در قالب یک تابع در مدل علی و به عنوان یک شرط بر روی مجموعه‌ی پیکربندی‌های ساختمان رویداد منتج شده از آن توصیف شود. این مساله کار کردن با این روش را برای کاربر سخت می‌کند. راه حل مناسب ارائه یک منطق در سطح زبان است که به کاربر اجازه‌ی توصیف رفتار ناامن یا در راه‌حلی بهتر اجازه‌ی توصیف ویژگی مورد نظر در قالب یک منطق را بدهد.

۴.۵ کارهای آینده

۱.۴.۵ ترکیب علت

در [۶] نویسندگان ثابت کرده‌اند که بر اساس تعریف آن‌ها علیت قابل ترکیب کردن در پردازیهایی که ارتباط همگام دارند وجود ندارد. یکی از انگیزه‌های اولیه‌ی این پژوهش دست‌یافتن به تعریفی از علت واقعی HP بود که بتوان علت خطا را در بخش‌های کوچک‌تر سیستم جستجو کرد. بنابراین یکی از مهم‌ترین کارهایی که در ادامه‌ی این پژوهش در نظر گرفته شده است اثبات قابلیت ترکیب علیت در پردازیهای همگام است.

۲.۴.۵ سنتز تعمیر

از علت‌های پیدا شده در سیستم برای بروز خطا می‌توان برای سنتز تعمیر مناسب برای سیستم استفاده کرد. به طور خاص در این روش که علت خطا در ساختارهای سیستم جستجو می‌شود سنتز کردن تعمیر فرآیند ساده‌تری خواهد بود.

فصل ۶

مروری بر کارهای پیشین

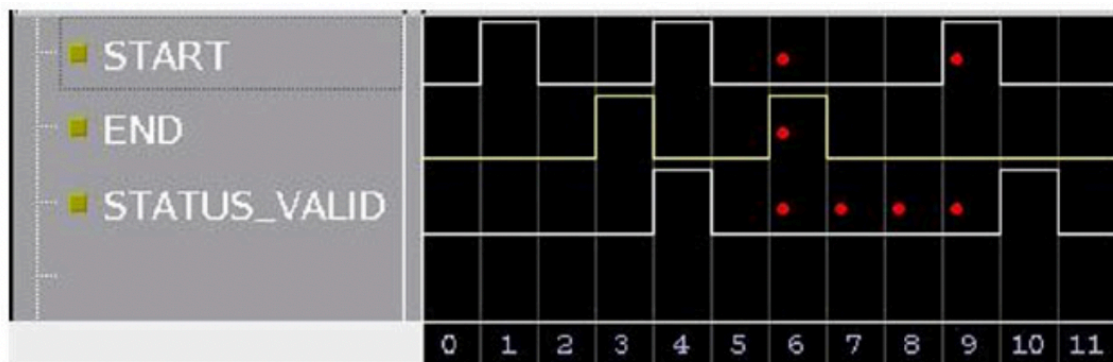
توضیح خطا^۱ و متمرکز کردن خطا^۲ روش‌هایی هستند که فرآیند رفع ایراد نرم‌افزار را تسهیل می‌کنند. توضیح خطا روش‌هایی را شامل می‌شود که به کاربر کمک می‌کند که با استفاده از یک مثال نقض یا یک دنباله اجرای سیستم به ماهیت خطا و در نتیجه روش اصلاح خطا پی ببرد. در روش‌های متمرکز کردن خطا هدف مشخص کردن بخشی از سیستم است که عامل خطا بوده و امکان اندازه‌گیری کمی و مقایسه آن با دیگر بخش‌ها وجود دارد [۱۰]. یکی از روش‌های توضیح خطا پیدا کردن علت خطا بر اساس استدلال مبتنی بر خلاف واقع که توسط لوئیس در [۱۸] ارائه شده می‌باشد که در پژوهش‌هایی مانند [۲۲، ۱۰، ۱۱] استفاده شده است. هالپرن و پرل در [۱۲] تعریفی مبتنی بر استدلال خلاف واقع لوئیس ارائه کردند که توانسته است برخی از مشکلات تعریف لوئیس در پیدا کردن علت در سناریوهای پیچیده را بر طرف کند. در ادامه برخی از پژوهش‌هایی که از تعریف هالپرن و پرل برای توضیح خطا استفاده کرده‌اند را مورد بررسی قرار می‌دهیم.

۱.۶ تخمین پوشش

در [۷] نویسندگان از مدل HP برای تخمین میزان پوشش سیستم توسط یک توصیف در فرآیند واریسی مدل استفاده کرده‌اند. معیارهای پوشش معمولاً در فرآیند تست سیستم استفاده می‌شوند و مشخص‌کننده درصدی از

^۱Fault Explanation

^۲Fault Localization



شکل ۱.۶: رابط کاربری ابزار PE RuleBase

اجزا یا حالت‌های سیستم هستند که توسط مجموعه‌ی تست‌ها مورد استفاده یا بازدید قرار می‌گیرند. در فرآیند واریسی مدل همه‌ی حالت‌های سیستم بررسی می‌شوند بنابراین در این شرایط اگر تغییر یک حالت منجر به نقض ویژگی توصیف شده شود این حالت پوشش داده شده توسط ویژگی تعریف می‌شود. با استفاده از مفهوم مسئولیت^۳ که در [۱۳] تعریف شده است، نویسندگان این پژوهش به جای در نظر گرفتن مقدار ۰ و ۱ برای پوشیده شدن یا نشدن از درجه‌ی مسئولیت استفاده می‌کنند. همانطور که مشخص است این پژوهش به اصلاح و بهبود توصیف ویژگی کمک می‌کند و نه پیدا کردن علت خطا.

۲.۶ علت خطا در مثال نقض

در [۲] نویسندگان سیستم را به صورت یک سیستم انتقال در نظر می‌گیرند که هر حالت یک نگاهت از یک مجموعه‌ی متغیرهای بولی به مقادیر درست و غلط است. در این پژوهش با استفاده از تعریف علت واقعی در یک مثال نقض یک ویژگی توصیف شده در LTL^۴ یک دوتایی متغیر و حالت به عنوان علت واقعی در نظر گرفته می‌شود. در همین پژوهش یک الگوریتم تقریبی برای پیدا کردن همه‌ی علت‌ها در یک مثال نقض داده شده ارائه شده است و ابزاری برای نمایش این علت‌ها به صورت گرافیکی به کاربر توسعه داده شده و در ابزار درستی‌سنجی PE RuleBase متعلق به IBM گنجانده شده است. تصویر ۱.۶ رابط کاربری ابزار PE RuleBase را پس از

^۳Responsibility^۴Linear Temporal Logic

پیدا کردن یک مثال نقض برای ویژگی زیر نشان می‌دهد:

$$G((\neg START \wedge \neg STATUS_{VALID} \wedge END) \rightarrow [\neg START \wedge STATUS_{VALID}])$$

در این تصویر نقاط قرمز علت‌های واقعی هستند که با الگوریتم تقریبی پیاده‌سازی شده پیدا شده‌اند. این پژوهش یکی از کاربردی‌ترین استفاده‌ها از توضیح خطا و پیدا کردن علت خطا را نشان می‌دهد. در این پژوهش سعی شده است تا علت خطا در یک مثال نقض پیدا شود و به همین دلیل مقدار متغیرها در حالت‌ها به عنوان علت پیدا می‌شوند در حالی که در پژوهش جاری هدف پیدا کردن علت خطا در کل سیستم است و در واقع ساختارهای سیستم، مثلاً وجود یا عدم وجود روابط تعارض یا فعال‌سازی به عنوان علت خطا پیدا می‌شوند.

۳.۶ چک کردن علیت

در پژوهش [۱۷] نویسندگان تعریفی از علت واقعی که الهام گرفته از تعریف HP است ارائه می‌کنند و الگوریتم آن‌ها بر اساس این تعریف در حین اجرای فرآیند واریسی مدل علت‌ها را پیدا کرده و در نتیجه در انتهای واریسی مدل اگر سیستم ویژگی مورد نظر را نقض کرد به جای برگرداندن یک مثال نقض، رویدادهایی که علت رخداد خطا بوده‌اند را بر می‌گرداند. در این پژوهش یک منطق برای توصیف یک دنباله از رویداد عملیات‌های سیستم ارائه شده است و فرمول‌های این منطق به عنوان علت خطا در نظر گرفته می‌شوند. این پژوهش هم همانند [۲] سعی بر پیدا کردن همه‌ی علت‌های بروز خطا دارد و علت‌ها عملاً دنباله‌هایی از اجرای سیستم هستند. تفاوت اصلی این کار با پژوهش جاری در این است که در این پژوهش علت خطا در رفتارهای سیستم جستجو می‌شود در حالی که در پژوهش جاری علت خطا در میان عناصر ساختاری سیستم جستجو می‌شود. این روش تنها برای ویژگی‌های دسترس‌پذیری ارائه شده است. در [۳] نویسندگان این روش را برای ویژگی‌های دلخواه توصیف شده توسط LTL تعمیم دادند.

۴.۶ علت واقعی در خودکاره‌های زمان‌دار

در [۱۵] نویسندگان از تعریف HP برای پیدا کردن علت خطا در خودکاره‌های زمان‌دار استفاده کرده‌اند. در درستی سنجی خودکاره‌های زمان‌دار یک ابزار واریسی مدل بلا درنگ نقض ویژگی را در قالب یک رد تشخیصی زمان‌دار^۵ که در واقع یک مثال نقض است بر می‌گرداند. یک TDT در واقع یک دنباله متناوب از انتقال تاخیر^۶ و انتقال عملیات^۷ ها است که در آن مقدار تاخیرها به صورت سمبلیک مشخص شده‌اند. هدف این پژوهش پیدا کردن مقادیری یا دامنه‌ای از مقادیر برای این تاخیرهای سمبلیک است که بروز خطا را اجتناب ناپذیر می‌کنند یا به عبارت دیگر علت واقعی هستند. در این پژوهش اما به صورت مستقیم از تعریف HP استفاده نشده است و بر اساس آن تعریفی برای علت واقعی نقض ویژگی در یک TDT بیان شده است.

۵.۶ چارچوب علیت بر اساس رد سیستم

در [۹] نویسندگان این مساله را مطرح می‌کنند که تعریف ارائه شده توسط هالپرن و پرل ذاتاً یک مدل بر اساس منطق گزاره‌ای است و به همین دلیل برای درستی سنجی پرده‌ها ایده‌آل نیست. در این پژوهش یک فرمالیسم و تعریف جدید برای علیت بر اساس تعریف HP ارائه می‌شود که در آن از ردهای سیستم به جای متغیرها در مدل HP استفاده می‌شود و امکان ترکیب چند مدل با یکدیگر را فراهم می‌کند.

۶.۶ استدلال مبتنی بر علیت در HML

در [۵] نویسندگان از مفهوم استدلال مبتنی در سیستم انتقال برچسب‌دار و HML^۸ [۱۴] استفاده کرده‌اند. در این پژوهش سیستم با استفاده از یک سیستم انتقال برچسب‌دار مدل می‌شود و رفتار ناامن توسط یک فرمول در قالب HML توصیف می‌شود. سپس یک تعریف جدید که برگرفته شده از تعریف HP است با استفاده از این

^۵Timed Diagnostic Trace

^۶Transition Delay

^۷Delay Transition

^۸Hennesy Milner Loigc

مدل‌ها برای علت واقعی بیان می‌شود. در این تعریف از مفهومی به نام عدم‌وقوع^۹ رویدادها که پیش‌تر در [۱۷] مطرح شده بود استفاده می‌شود. شهود کلی مفهوم عدم‌وقوع در علت این است که در کنار اینکه رخ دادن برخی از رویدادها منجر به خطا می‌شود، رخ ندادن رویدادها هم می‌تواند به عنوان علت در نظر گرفته شود. در تعریف ارائه شده در این پژوهش مجموعه‌ای از محاسبه^{۱۰} های سیستم به عنوان علت برقراری یک فرمول HML در سیستم که رفتار ناامن را توصیف می‌کند تعریف می‌شود. هر محاسبه شامل یک دنباله از عملیات‌های سیستم در کنار تعدادی عملیات دیگر، که عدم‌وقوع آن‌ها هم جزئی از علت است، در نظر گرفته می‌شوند. به عبارت دیگر یک محاسبه را می‌توان شامل دو جز در نظر گرفت. جز اول یک اجرای سیستم است که منجر به خطا می‌شود. جز دوم مجموعه‌ای از اجراهای سیستم است که منجر به خطا نمی‌شوند و حاصل جایگذاری برخی از عملیات‌ها در جز اول این محاسبه هستند. عملیات‌های جایگذاری شده عملیات‌هایی هستند که عدم‌وقوع آن‌ها به عنوان علت بروز خطا در نظر گرفته می‌شود. در این تعریف علت واقعی به گونه‌ای تعریف شده است که محاسباتی که منجر به فعال شدن فرمول HML در سیستم می‌شوند به عنوان علت در نظر گرفته می‌شوند. در این تعریف شروطی مشابه با شروط موجود در تعریف HP در نظر گرفته شده است. در [۶] نویسندگان تعریف خود را بهبود دادند تا تطابق بیشتری با تعریف HP داشته باشد. علاوه بر این در این پژوهش ثابت شده است که این تعریف از علت در سیستم‌هایی که ارتباط همگام شده دارند قابل ترکیب نیست ولی در حالتی که سیستم‌ها ارتباط همگام نداشته باشند امکان ترکیب یا شکستن آن وجود دارد. نتایج حاصل از این پژوهش یکی از انگیزه‌های اصلی پژوهش جاری بود برای اینکه با انتخاب یک معنا یا تعریف علت متفاوت امکان ترکیب آن برای سیستم‌های همگام شده بررسی شود. در ادامه به بررسی شباهت‌ها و تفاوت‌های این پژوهش و پژوهش جاری می‌پردازیم. اولاً در این پژوهش تعریف جدیدی از علت واقعی ارائه شده است که در حالی که در پژوهش جاری مستقیماً از تعریف ارائه شده در [۱۲] استفاده شده است. در پژوهش جاری تمرکز بر پیدا کردن یک علت برای بروز خطا در سیستم است در حالی که در این پژوهش همه‌ی علل خطا مورد بررسی قرار می‌گیرند. پژوهش جاری علل خطا را در ساختارهای سیستم جستجو می‌کند در حالی که این پژوهش در میان رفتارهای سیستم به دنبال علل خطا می‌گردد.

^۹Non-Occurrence^{۱۰}Computation

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. [8](#)
- [2] Beer, Ilan, Ben-David, Shoham, Chockler, Hana, Orni, Avigail, and Trefler, Richard. Explaining counterexamples using causality. *Formal Methods in System Design*, 40(1):20–40, Feb 2012. [54](#), [58](#), [59](#)
- [3] Caltais, Georgiana, Guetlein, Sophie Linnea, and Leue, Stefan. Causality for general ltl-definable properties. *Electronic Proceedings in Theoretical Computer Science*, 286:1–15, Jan 2019. [59](#)
- [4] Caltais, Georgiana, Hojjat, Hossein, Mousavi, Mohammad, and Tunc, Hunkar Can. Dynetkat: An algebra of dynamic networks. *arXiv preprint arXiv:2102.10035*, 2021. [15](#), [36](#)
- [5] Caltais, Georgiana, Leue, Stefan, and Mousavi, Mohammad Reza. (de-)composing causality in labeled transition systems. *Electronic Proceedings in Theoretical Computer Science*, 224:10–24, Aug 2016. [60](#)
- [6] Caltais, Georgiana, Mousavi, Mohammad Reza, and Singh, Hargurbir. Causal reasoning for safety in hennessy milner logic. *Fundamenta Informaticae*, 173(2–3):217–251, Mar 2020. [54](#), [55](#), [61](#)
- [7] Chockler, Hana, Halpern, Joseph Y., and Kupferman, Orna. What causes a system to satisfy a specification? *ACM Transactions on Computational Logic*, 9(3):1–26, Jun 2008. [57](#)
- [8] Foerster, Klaus-Tycho, Schmid, Stefan, and Vissicchio, Stefano. Survey of consistent software-defined network updates. *IEEE Communications Surveys & Tutorials*, 21(2):1435–1461, 2018. [3](#), [8](#), [45](#)

- [9] Gössler, Gregor and Le Métayer, Daniel. A general trace-based framework of logical causality. In *International Workshop on Formal Aspects of Component Software*, pages 157–173. Springer, 2013. [60](#)
- [10] Groce, Alex, Chaki, Sagar, Kroening, Daniel, and Strichman, Ofer. Error explanation with distance metrics. *International Journal on Software Tools for Technology Transfer*, 8(3):229–247, 2006. [57](#)
- [11] Groce, Alex and Visser, Willem. What went wrong: Explaining counterexamples. In *International SPIN Workshop on Model Checking of Software*, pages 121–136. Springer, 2003. [57](#)
- [12] Halpern, Joseph Y. and Pearl, Judea. Causes and explanations: A structural-model approach, part i: Causes. *arXiv:cs/0011012*, Nov 2005. arXiv: cs/0011012. [5](#), [23](#), [25](#), [28](#), [53](#), [57](#), [61](#)
- [13] Halpern, Joseph Y. and Pearl, Judea. Causes and explanations: A structural-model approach. part ii: Explanations. *The British journal for the philosophy of science*, 56(4):889–911, 2005. [58](#)
- [14] Hennessy, Matthew and Milner, Robin. On observing nondeterminism and concurrency. In *International Colloquium on Automata, Languages, and Programming*, pages 299–309. Springer, 1980. [60](#)
- [15] Kölbl, Martin, Leue, Stefan, and Schmid, Robert. Dynamic causes for the violation of timed reachability properties. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 127–143. Springer, 2020. [60](#)
- [16] Kozen, Dexter. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19(3):427–443, 1997. [8](#)
- [17] Leitner-Fischer, Florian and Leue, Stefan. Causality checking for complex system models. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, page 248–267. Springer, 2013. [54](#), [59](#), [61](#)
- [18] Lewis, David. *Counterfactuals*. Wiley, 1973. [57](#)
- [19] 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. [8](#)
- [20] Reitblatt, Mark, Foster, Nate, Rexford, Jennifer, Schlesinger, Cole, and Walker, David. Abstractions for network update. *ACM SIGCOMM Computer Communication Review*, 42(4):323–334, 2012. [41](#), [49](#)

- [21] Winskel, Glynn. *Event structures*, volume 255 of *Lecture Notes in Computer Science*, page 325–392. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987. [20](#), [31](#)
- [22] Zeller, Andreas. *Why programs fail: a guide to systematic debugging*. Elsevier, 2009. [57](#)

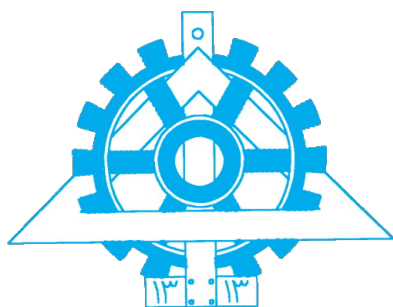
واژه‌نامهٔ فارسی به انگلیسی

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