**دانشگاه تهران**

**پردیس دانشکده های فنی**

**دانشکده مهندسی برق و کامپیوتر**

توضیح خطا در شبکه های مبتنی بر نرم افزار بااستفاده از استدلال مبتنی بر علیت

**پایان نامه برای دریافت درجهٔ کارشناسی ارشد در رشتهٔ مهندسی کامپیوتر**

**گرایش نرم افزار**

**امیرحسین صیحانی**

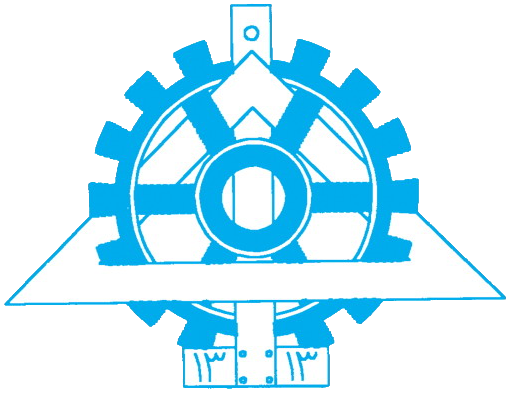
استاد راهنما

**دکتر حسین حجت**

استاد مشاور

# دکتر محمدرضا موسوی

شهریور ۱۰۴۱

**دانشگاه تهران**

**پردیس دانشکده های فنی**

**دانشکده مهندسی برق و کامپیوتر**

# گواهی دفاع از پایان نامه کارشناسی ارشد

هیأت داوران پایان نامهٔ کارشناسی ارشد آقای / خانم امیرحسین صیحانی به شمارهٔ دانشجویی ۸۱۰۱۹۸۱۹۸ دررشتهٔ مهندسی کامپیوتر - گرایش نرم افزار را در تاریخ ............ با عنوان »توضیح خطا در شبکه های مبتنی بر

نرم افزار با استفاده از استدلال مبتنی بر علیت«

به عدد به حروفبا نمرهٔ نهایی

.

ارزیابی کرد

و درجهٔ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **امضا** | **دانشگاه یا مؤسسه** | **دانشگاهی** | **نام و نام خانوادگی** | **مشخصات هیأت**  **داوران** | **ردیف** |
|  | دانشگاه تهران | استادیار | دکتر حسین حجت | استاد راهنما | ۱ |
|  | کینگز کالج لندن | استاد | دکتر محمدرضاموسوی | استاد مشاور | ۲ |
|  | دانشگاه تهران |  |  | استاد داور داخلی | ۳ |
|  |  |  |  | استاد مدعو | ۴ |
|  | دانشگاه تهران |  |  | نمایندهٔ تحصیلات  تکمیلی دانشکده | ۵ |

**مرتبهٔ**

نام و نام خانوادگی معاون تحصیلات تکمیلی و نام و نام خانوادگی معاون آموزشی و تحصیلاتپژوهشی دانشکده / گروه: تکمیلی پردیس دانشکده های فنی:

تاریخ و امضا: تاریخ و امضا:

## تعهدنامهٔاصالتاثر

باسمه تعالی

اینجانب امیرحسین صیحانی تأیید می کنم که مطالب مندرج در این پایان نامه حاصل کار پژوهشی اینجانباست و به دستاوردهای پژوهشی دیگران که در این نوشته از آن ها استفاده شده است مطابق مقررات ارجاع گردیدهاست. این پایان نامه قبلاً برای احراز هیچ مدرک هم سطح یا بالاتری ارائه نشده است.

نام و نام خانوادگی دانشجو: امیرحسین صیحانیتاریخ و امضای دانشجو:

**کلیهٔ حقوق مادی و معنوی این اثر**

**متعلّق به دانشگاه تهران است.**

# چکیده

جایگزینی شبکه های کامپیوتری سنتی با شبکه ها مبتنی بر نرم افزار۱ باعث شده است تا استفاده از روش ها وابزار های مبتنی بر روش های صوری۲ برای درستی سنجی۳ این شبکه ها تسهیل شود. با اینکه درستی سنجی لازمه یرفع ایراد در سیستم ها است، اما پس از مشخص شدن وجود خطا در سیستم، پیدا کردن دلیل این مساله که چراسیستم دچار خطا شده است همچنان بر عهده ی کاربر است و لازم است که او با تحلیل و بررسی سیستم علتمشکل را پیدا کند و ایراد سیستم را بر طرف کند. ابزار های درستی سنجی در صورتی که سیستم مطابق انتظار رفتارنکند یک مثال نقض یا گواهی برای اثبات این موضوع به کاربر ارائه می کنند اما این مدارک به تنهایی و بدون پردازشبیشتر اطلاعات کافی از چرایی مشکل در اختیار نمی گذارند.

توضیح پدیده ها در متون فلسفه قرن ها مورد مطالعه قرار گرفته است و نتایج این مطالعات در قالب اصولاستدلال مبتنی بر خلاف واقع۴ تجمیع شده است. هالپرن۵ و پرل۶ یک فرمولاسیون ریاضی برای پیدا کردن علتواقعی۷ بر اساس استدلال مبتنی بر خلاف واقع ارائه کرده اند که در پژوهش های متعددی برای ارائه توضیح درمورد خطای رخ داده در سیستم و پیدا کردن علت واقعی آن مورد استفاده قرار گرفته است.

در این پژوهش از مفهوم علت واقعی ارائه شده توسط هالپرن و پرل برای توضیح خطا در شبکه های مبتنی برنرم افزار استفاده می شود. به صورت دقیق تر در این پژوهش یک مدل علّی۸ ارائه می شود که با استفاده از آن می تواندر مورد ساختارهای شبکه، مانند وجود هم روندی یا ترتیب میان به روز رسانی های شبکه، برای پیدا کردن علتواقعی رفتار نا امن شبکه استدلال کرد.

در این پژوهش برای توصیف نرم افزار شبکه از زبان نت کت پویا۹ استفاده می شود. نت کت پویا یک سطح بالا برنامه نویسی شبکه است که بر پایه نت کت۱۰ بنا شده و با وجود اینکه مینیمال بودن آن را حفظ کرده امکانتوصیف به روز رسانی های پویای شبکه را فراهم می کند. در این پژوهش از ساختمان رویداد۱۱ به عنوان مدل

1Software-Defined Network

2Formal Methods

3Verification

4Counterfactual Reasoning

5Joseph Y. Halpern

6Judea Pearl

7Actual Cause

8Casual Model

9DyNetKAT

10NetKAT

11Event Structure

توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت

معنایی۱۲ برنامه های نت کت پویا استفاده می شود. ساختمان رویداد امکان توصیف صریح هم روندی را فراهممی کند که این موضوع سبب می شود چنین روابطی میان پردازه ها را هم بتوان به عنوان علت خطا در نظر گرفت،امری که با استفاده از مدل های برگ برگ شده۱۳ امکان پذیر نیست. روش ارائه شده در این پژوهش برای پیدا کردنعلت نقض چند ویژگی مطرح شبکه، مثلا نبود دور، مورد استفاده قرار گرفته است و بررسی این مساله ها نشانمی دهد که علت های پیدا شده با شهود موجود از علت خطا در سیستم تطابق دارند.

**واژگان کلیدی** استدلال مبتنی بر علیت، ساختمان رویداد، درستی سنجی، شبکه های مبتنی بر نرم افزار

12Semantic Model

13Interleaving ده

# فهرست مطالب

|  |  |
| --- | --- |
| **مقدمه ۱** | **فصل ۱:** |
| ساختار فصل ها ۵ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۱ .۱ |
| **تعاریف و دانش پیش زمینه ۷** | **فصل ۲:** |
| مقدمه ۷ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۱ .۲ |
| شبکه های مبتنی بر نرم افزار ۷ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۲ .۲ |
| نت کت ۸ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  ۱ .۳ .۲ دستور زبان نت کت ............................ ۸  ۲ .۳ .۲ مدل معنایی نت کت ........................... ۹  ۳ .۳ .۲ توصیف رفتار شبکه با نت کت ....................... ۱۲  ۴ .۳ .۲ درستی سنجی برنامه های نت کت ...................... ۱۴ | ۳ .۲ |
| نت کت پویا ۱۷ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  ۱ .۴ .۲ دستور زبان نت کت پویا ......................... ۱۷  ۲ .۴ .۲ معنای عملیاتی نت کت پویا ........................ ۱۸  ۳ .۴ .۲ توصیف برنامه ها در نت کت پویا ...................... ۲۰ | ۴ .۲ |
| ساختمان رویداد ۲۲ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۵ .۲ |
| مدل علّی ۲۵ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۶ .۲ |

۱ .۶ .۲ علت واقعی ............................... ۲۸

۲ .۶ .۲ پیدا کردن علت واقعی در مسائل ...................... ۲۹

۳ .۶ .۲ مدل تعمیم یافته ............................. ۳۱

آ

فهرست مطالب توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت

۴ .۶ .۲ علت واقعی بدون شرط .......................... ۳۲

|  |  |
| --- | --- |
| **مروری بر کا رهای پیشین ۳۳** | **فصل ۳:** |
| علت خطا در مثال نقض ۳۳ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۱ .۳ |
| علت خطا در سیستم های قابل تنظیم ۳۵ . . . . . . . . . . . . . . . . . . . . . . . . . | ۲ .۳ |
| علت خطا در پروتکل های امنیتی ۳۵ . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۳ .۳ |
| چک کردن علیت ۳۶ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۴ .۳ |
| علت واقعی در خودکاره های زمان دار ۳۶ . . . . . . . . . . . . . . . . . . . . . . . . . | ۵ .۳ |
| چارچوب علیت بر اساس رد سیستم ۳۷ . . . . . . . . . . . . . . . . . . . . . . . . . | ۶ .۳ |
| استدلال مبتنی بر علیت درLMH ۳۷ . . . . . . . . . . . . . . . . . . . . . . . . . | ۷ .۳ |
| جمع بندی ۳۸ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۸ .۳ |
| **روش و راه حل پیشنهادی ۴۱** | **فصل ۴:** |
| مقدمه ۴۱ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۱ .۴ |
| مدل معنایی عبارات نت کت پویا در قالب ساختمان رویداد ۴۱ . . . . . . . . . . . . . . .  ۱ .۲ .۴ معنای عبارات نت کت پویای نرمال .................... ۴۵ | ۲ .۴ |
| مدل علّی برای ساختمان رویداد ۴۷ . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۳ .۴ |
| پیدا کردن علت خطا در نت کت پویا ۵۰ . . . . . . . . . . . . . . . . . . . . . . . . . | ۴ .۴ |
| بررسی چند ویژگی شبکه ۵۰ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۵ .۴ |
| لیست سیاه ۵۱ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۶ .۴ |
| نبود دور ۵۵ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۷ .۴ |
| نبود سیاه چاله ۵۸ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۸ .۴ |
| **جمع بندی و کا رهای آینده ۶۳** | **فصل ۵:** |
| جمع بندی کار های انجام شده ۶۳ . . . . . . . . . . . . . . . . . . . . . . . . . . . . | ۱ .۵ |
| نوآوری ها و دستاورد ها ۶۴ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  ۱ .۲ .۵ جستجو در ساختار ............................ ۶۴ | ۲ .۵ |

ب

توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت فهرست مطالب

|  |  |
| --- | --- |
| ۲ .۲ .۵ استفاده از ساختمان رویداد به عنوان مدل معنایی ............... ۶۴  ۳ .۲ .۵ استفاده مستقیم از تعریف علت ...................... ۶۴ |  |
| محدودیت ها ۶۵ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  ۱ .۳ .۵ پیچیدگی زمانی ............................. ۶۵  ۲ .۳ .۵ توصیف خطا در سطح مدل علی ...................... ۶۵  ۳ .۳ .۵ استدلال در مورد یک علت ........................ ۶۵ | ۳ .۵ |
| کار های آینده ۶۶ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  ۱ .۴ .۵ ترکیب علت ............................... ۶۶  ۲ .۴ .۵ سنتز تعمیر ............................... ۶۶  ۳ .۴ .۵ مقایسه و رتبه بندی علت ها ........................ ۶۶ | ۴ .۵ |
| **۶۹** | **مراجع** |

پ

**فصل ۱مقدمه**

در شبکه های کامپیوتری رفتار اجزای شبکه را می توان در یکی از این دو دسته قرار داد: سطح کنترل۱ و سطحداده۲. در سطح کنترلتصمیم گیری در مورد چگونگی رسیدگی به ترافیک شبکه انجاممی شود، مثلا چه پورت هاییباید باز شوند یا چه نوع بسته هایی اجازه ی عبور دارند. در سطح داده، رفتار هایی که در سطح کنترل در مورد آن هاتصمیم گیری شده است صرفا اجرا می شوند. مثلا باز کردن پورت ها یا عبور دادن بسته هایی از یک نوع خاصرفتار هایی هستند که در سطح داده طبقه بندی می شوند. در شبکه های کامپیوتری فعلی رفتار های این دو سطح دراجزای شبکه تجمیع شده اند. به همین دلیل یک شبکه ی کامپیوتری عملا یک سیستم توزیع شده شامل برنامه هاییاست که برای هر یک از اجزای شبکه به شکل مجزا نوشته شده است و این مساله باعث پیچیدگی شبکه و مدیریتآن می شود [۲۴]. شبکه های مبتنی بر نرمافزار۳ برای حل این مشکل از یک نرم افزار متمرکز برای کل شبکه استفادهمی کنند. به طور دقیق تر، در شبکه های مبتنی بر نرم افزار، رفتار های سطح کنترل و داده از یکدیگر جدا می شوند.در نتیجه ی این جداسازی اجزای شبکه مانند سوییچ ها یا روترها دستگاه های ساده ای در نظر گرفته می شوند کهتنها رفتار های سطح داده دارند و رفتار های سطح کنترل توسط یک نرم افزار متمرکز انجام می شود. بنابراین، در یکشبکه ی مبتنی بر نرم افزار، مدیر شبکه یک برنامه برای مدیریت کل شبکه می نویسد و دیگر نیازی به برنامه نویسیبرای تک تک اجزای شبکه ندارد.

با توجه به نقش حیاتی شبکه ها در سیستم های کامپیوتری، اطمینان از عملکرد درست آن ها از اهمیت بالایی

1Control Plane

2Data Plane

3Software Defined Network

۱

فصل ۱: مقدمه توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیتبرخوردار است [۱۴]. روش های صوری۴ مجموعه ای از زبان های مبتنی بر ریاضی، تکنیک ها و ابزار هایی برایتوصیف و درستی سنجی سیستم های سخت افزاری و نرم افزاری هستند [۱۲]. شبکه های مبتنی بر نرم افزار با متمرکزکردن رفتار کنترل کننده ی شبکه و ساده تر کردن اجزای دیگر شبکه امکان به کار گیری چنین روش هایی را تسهیلکرده اند. روش های متعددی مانند [۱، ۲۱، ۳۱، ۳۶] برای درستی سنجی شبکه های مبتنی بر نرم افزار ارائه شده اند.

] OpenFlow۲۷] مطرح ترین رابط برنامه نویسی۵ برای شبکه های مبتنی بر نرم افزار است. اما برنامه هاینوشته شده باOpenFlow معمولا سطح پایین هستند و کار کردن با آن ها برای کاربر معمولا دشوار است. به همیندلیل زبان های برنامه نویسی متعددی مانند [۱۵، ۳۳، ۲۸، ۳۴، ۲۹، ۲] ارائه شده اند که با استفاده ازwolFnepO امکان برنامه نویسی برای شبکه های مبتنی بر نرم افزار در سطح بالاتر را فراهم می کنند. نت کت یکی از این زبان هااست۶ [۲] که بر پایه یTAK] ۲۳] بنا شده است. استفاده ازKAT و داشتن یک سیستم معادلاتی صحیح۷ وکامل۸ باعث می شود تا اثبات درستی برنامه ها در نت کت را بتوان با روش های جبری و اثبات تساوی برنامه هایمختلف توصیف شده در این زبان انجام داد. نت کت پویا۹[۷] برای بهبود برخی از قابلیت های نت کت ارائه شدهاست که از جمله این قابلیت ها میتوان به امکان توصیف به روز رسانی های شبکه و استدلال در مورد چندین بستهدر شبکه اشاره کرد.

در درستی سنجی نرمافزار با روش های صوری یک مدل از سیستم و رفتار مورد انتظار آن در قالب یک مدلریاضی توصیف می شود و با روش هایی مبتنی بر الگوریتم، مانند وارسی مدل۱۰ [۱۱]، می توان ثابت کرد که سیستمبا رفتار مورد انتظار تطابق دارد یا خیر. یکی از مهم ترین ویژگی های الگوریتم های درستی سنجی امکان تولیدمثال نقض ۱۱ یا گواهی۱۲ برای اثبات نقض رفتار مورد انتظار توسط سیستم است. این مثال نقض یا گواهی ها با اینکه میتوانند در مورد رفتار سیستم توضیح دهند ولی درک درستی از این که چرا ویژگی مورد نظر در سیستمنقض شده است به دست نمی دهند. به دست آوردن چنین درکی از اینکه چرا سیستم به درستی و مطابق انتظار کارنمی کند به آنالیز و تحلیلی فراتر نیاز دارد. استفاده از علیت۱۳ و پیدا کردن علت خطا یکی از راهکارها برای به دست

4Formal Methods

5Application Programming Interface

6NetKAT

7Sound

8Complete

9DyNetKAT

10Model Checking

11Counterexample

12Certificate

13Causality

۲

توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت فصل ۱: مقدمه

*a*

*b*

*c*

*d*

*α*

*β*

شکل ۱ .۱: به روز رسانی*α*  مسیر سبز پر رنگ را با مسیر خط چین جایگزین می کند. به روز رسانی*β*  مسیر نارنجیپر رنگ را با مسیر خط چین جایگزین می کند.

آوردن درک بیشتر از مشکل سیستم است. مفهوم علیت و مبانی آن قرن ها در متون فلسفه مورد مطالعه قرار گرفتهو استدلال مبتنی بر خلاف واقع۱۴ روشی است که در نهایت برای پیدا کردن علت واقعی مورد استفاده قرار گرفتهاست. با وجود اینکه چنین نظریه ای مدت ها پیش مورد توافق قرار گرفته است، فرمولاسیون دقیق ریاضی آن درسال های اخیر توسط هالپرن۱۵ و پرل۱۶ در [۱۸] ارائه شده است [۴]. این مدل از علت واقعی در چندین پژوهشمانند [۶، ۹، ۲۵، ۵] مورد استفاده قرار گرفته است تا علت واقعی خطا در سیستم پیدا شود. هدف پژوهش جاریاستفاده از این مدل علیت برای پیدا کردن علت نقض ویژگی در برنامه های توصیف شده با زبان نت کت پویا است.به عنوان مثال شبکه ی شکل ۱ .۱ را در نظر بگیرید. در این شبکه یک مسیر از*a*  به*d*  وجود دارد و نیاز است تا اینمسیر به روز رسانی شود تا ابتدا از سوییچ*c*  عبور کند. برای این منظور دو به روز رسانی*α*  و*β*  در شبکه انجام میشوند که به ترتیب مسیرهای سبز و نارنجی پر رنگ را با مسیر های خط چین جایگزین می کنند. پس از اجرایهر دو به روز رسانی مسیر جدیدی از*a*  به*d*  ایجاد می شود. فرض کنید که به روز رسانی ها به گونه ای انجام شوندکه ابتدا*β*  و سپس*α*  انجام شود. در این حالت پس از اجرای اولین به روز رسانی یک دور۱۷ شامل سوییچ های*b*  و*c* به وجود می آید. نبود دور یکی از ویژگی های رایجی است که در شبکه های کامپیوتری مورد بررسی قرار می گیرد[۳۰]. بنابراین اگر برنامه ی این شبکه به گونه ای توصیف شده باشد که امکان انجام به روز رسانی ها به این شکل راداشته باشد، این شبکه ویژگی بدون دور بودن را نقض می کند. با توجه به ساده بودن این مثال در اینجا به سادگیمی توان دریافت که انجام شدن*β*  پیش*α*  علت خطا بوده است. هدف پژوهش جاری پیدا کردن چنین عواملی به

14Counterfactual Reasoning

15Joseph Y. Halpern

16Judea Pearl

17Loop

۳

فصل ۱: مقدمه توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیتعنوان علت خطا در شبکه های مبتنی بر نرم افزار است.

زبان نت کت پویا، چون بر پایه ی نت کت بنا شده است، در کنار حفظ ساختار مینیمال و ساده ی خود امکانتوصیف به روز رسانی های شبکه را هم فراهم می کند و با توجه به اینکه به روز رسانی های شبکه منشا مهمی برایبروز خطا در شبکه هستند، از این زبان در این پژوهش استفاده شده است. در این پژوهش سعی شده استتا رویکرد متفاوتی نسبت به پژوهش هایی مانند [۲۵، ۹، ۸] اتخاذ شود. اولا در این پژوهش روابط ساختاریعملیات ها، مثلا هم روندی یا تقدم و تاخر آن ها، به عنوان علت در نظر گرفته می شوند. ثانیا در این پژوهش بهصورت مستقیم از تعریف علت واقعی مطابق [۱۸] استفاده می شود. برای رسیدن به دو هدف ذکر شده از ساختمانرویداد۱۸ [۳۵] به عنوان مدل معنایی برنامه های توصیف شده در نت کت پویا استفاده شده است. ساختمان رویدادیکمدلمحاسباتی۱۹ برایپردازه هایهم رونداست. درمدل هایبرگ برگشده۲۰مانندسیستمانتقال۲۱ همروندیپردازه ها به صورت صریح توصیف نمی شود و با انتخاب غیرقطعی۲۲ بین ترتیب های ممکن اجرای آن ها جایگزینمی شود. اما ساختمان رویداد یک مدل غیر برگ برگ شده۲۳ است که در آن هم روندی پردازه ها به صورت صریحتوصیف می شود. استفاده از این مدل کمک می کند که هم روندی عملیات ها هم بتواند به عنوان علت خطا در نظرگرفته شود، امری که با استفاده از مدل های برگ برگ شده ممکن نیست.

به صورت خلاصه برای پیدا کردن علت خطا در یک برنامه ی نت کت پویا ابتدا، با استفاده از مدل معنایی،ساختمان رویداد معادل برنامه محاسبه می شود. سپس یک مدل علّی۲۴ بر اساس معادلات ساختاری۲۵ که در[۱۸] از آن استفاده شده است ساخته می شود. در نهایت با توصیف کردن رفتار نا امن۲۶ در مدل علّی و با استفادهاز تعریف علت واقعی هالپرن و پرل، علت واقعی رفتار نا امن پیدا می شود. در مثال شکل ۱ .۱ با استفاده از اینروش می توان نبود این شرط در سیستم که الزاما به روز رسانی*β*  پس از*α*  انجام شود را به عنوان علت واقعی بهوجود آمدن دور در شبکه معرفی کرد.

18Event Structure

19Computational Model

20Interleaving

21Transition System

22Non-Deterministic Choice

23Non-Interleaving

24Causal Model

25Strucutral Equations

26Unsafe Behavior

۴

توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت فصل ۱: مقدمه

## ۱ .۱ ساختار فصل ها

در فصل دوم تعاریف و دانش پیش زمینه ی مورد نیاز برای بقیه فصول بیان می شود. در فصل سوم مروری برکار های پیشین و مرتبط با این پژوهش انجام می شود. فصل چهارم روش ساخت یک مدل علّی از نقض ویژگی دریک برنامه ی توصیف شده در زبان نت کت پویا بیان می شود. سپس به کار گیری روش ارائه شده در این پژوهشبرای پیدا کردن علت خطا در چند دسته از ویژگی های رایج شبکه مورد بررسی قرار می گیرد. در این فصل بررسیمی شود که علت واقعی پیدا شده تا چه میزان با شهود موجود از مساله تطابق دارد و این فرمولاسیون تا چه حدموفق عمل می کند. در نهایت فصل پنجم شامل جمع بندی کار های انجام شده در این پژوهش و بحث در موردکاستی های آن و کار های پیش رو است.

۵

# فصل ۲تعاریف 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 بخشاست. ابتدا مفاهیم کلی شبکه های مبتنی بر نرم افزار بررسی می شوند. سپس زبان های نت کت و نت کت پویا کهزبان های مورد استفاده در این تحقیق هستند شرح داده می شوند. در ادامه تعاریف اولیه ساختمان رویداد۱ که بهعنوان مدل معنایی در این تحقیق استفاده می شود شرح داده می شود. در بخش آخر مدل علّی۲ که در این تحقیقبرای پیدا کردن علت واقعی خطا مورد استفاده قرار می گیرد توصیف شده است.

## ۲ .۲ شبکه های مبتنی بر نرم افزار

شبکه هایکامپیوترییکیاززیر ساخت هایحیاتیسیستم هایکامپیوتریهستند. باوجوداینکهنیازمندی هایاین شبکه ها بسیار بیشتر و پیشرفته تر نسبت به گذشته شده است تکنیک ها و متد های مدیریت و ساخت آن هاهمچنان مانند گذشته است. این مساله مدیریت آن ها را در استفاده های امروزی پیچیده و مستعد خطا می کند.حتی شرکت های بزرگی مانندAmazon Github، یاGoDaddy مرتبا مشکلاتی در شبکه های خود پیدا می کنند

1Event Structure

2Causal Model

۷

[۱۴]. شبکه های مبتنی بر نرم افزار یک پارادایم جدید برای طراحی و پیاده سازی شبکه های کامپیوتری هستند کهعلاوه بر اینکه مدیریت و درستی سنجی آن ها را با روش های اصولی تر امکان پذیر می کنند، انعطاف بالایی همدارند. به صورت خلاصه در یک شبکه ی مبتنی بر نرم افزار رفتار های کنترلی (تغییر و به روز رسانی قوانین ارسال۳) از عناصر شبکه مانند سوییچ ها یا روترها جدا می شوند و توسط یک کنترل کننده ی مرکزی انجام می شوند. بهعنوان مثال در زبانwolFnepO] ۲۷] رفتار سوییچ های شبکه تنها توسط تعدادی قانون به شکل تطبیق۴ و اجرا۵توصیف می شود. قوانینی به این شکل ابتدا بررسی می کنند که بسته با قانون تطابق داشته باشد و اگر تطابق وجودداشت عملیات توصیف شده در قانون اجرا می شود. با ساده شدن عناصر شبکه، تمامی به روز رسانی ها و تغییر هایلازم در شبکه توسط یک کنترل کننده مرکزی انجام می شود. متمرکز شدن رفتار کنترلی سیستم باعث می شود تااستدلال و درستی سنجی در مورد رفتار شبکه آسان تر شود.

## ۳ .۲ نت کت

نت کت ،یک زبان برای توصیف شبکه های مبتنی بر نرم افزار است [۲]. این زبان با وجود دستور زبان۶ ساده ایکه دارد، بر اساسTAK] ۲۳] بنا شده و به همین دلیل یک سیستم معادلاتی صحیح و کامل۷ دارد. این سیستممعادلاتی کمک می کند تا با استفاده از روش های جبری و اثبات تساوی برنامه های توصیف شده در این زبان بتواندر مورد آن ها استدلال کرد.

### **۱ .۳ .۲ دستور زبان نت کت**

در نت کت هر بسته به عنوان یک نگاشت از یک مجموعه از فیلد های*f*1*,f*2*,...,fn*  به اعداد طبیعی با تعدادارقام ثابت در نظر گرفته می شود. آی پی ۸ های مبدا و مقصد، نوع بسته و پورت ۹ های مبدا و مقصد مثال هایی از

3Forwarding Rule

4Match

5Action

6Syntax

7Sound and Complete

8IP

9Port

۸

*H*

1

*S*

*A*

*S*

*B*

*H*

2

۱

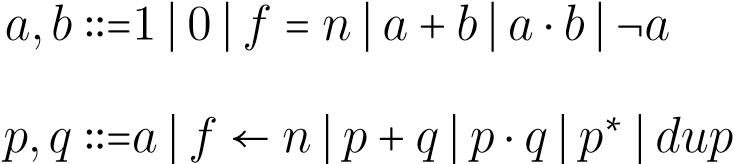
۲

۱

۲

شکل ۱ .۲: مثال شبکه

این فیلد ها هستند. دستور زبان نت کت به صورت زیر تعریف می شود:



در این گرامر عبارت های*a,b*  عملا معادل با تست های زبان کت۱۰ هستند. عبارت های*p,q*  عبارت های نت کترا تعریف می کنند که نسبت به دستور زبان کت جمله هایی به شکل*dup*  و*f* ← *n*  به آن اضافه شده است.برای مثال شبکه ی ۱ .۲ را در نظر بگیرید که شامل دو۱۱A و B و دو میزبان۱۲ است. هر سوییچ دو پورت داردکه با شماره های ۱ و۲ مشخص شده اند. با استفاده از عبارت نت کت زیر میتوانیم رفتار سوییچ های این شبکه راتوصیف کنیم:(۱ .۲) (2 ←*p* ≜ (*dst* = *H*1 ⋅ *pt* ← 1) + (*dst* = *H*2 ⋅ *pt*

این عبارت همه ی بسته هایی که مقصد آن ها میزبان ۱ باشد را به پورت ۱ و همه ی بسته هایی که مقصد آن ها میزبان۲ باشد را به پورت شماره ی ۲ می فرستد.

### **۲ .۳ .۲ مدل معنایی نت کت**

برای اینکه امکان استدلال در مورد مسیر های طی شده توسط یک بسته در شبکه وجود داشته باشد،از مفهومیبه نام تاریخچه ی بسته۱۳ استفاده می شود. هر تاریخچه ی بسته ،یک دنباله از بسته ها است که بسته نخست دنباله،

10KAT

11Switch

12Host

13Packet History

۹

به عنوان بسته ی فعلی در نظر گرفته می شود. به صورت شهودی عبارت های ۱ و ۰ به ترتیب به معنای ارسال۱۴ ورها کردن۱۵*f* ← *n* بدون شرط بسته هستند. عبارت*f* = *n*  در صورتی بسته را عبور می دهد که مقدار فیلدf آن برابر باn باشد. عبارت مقدارn را به فیلدf بسته اختصاص می دهد. عبارتهای*dup*  باعث میشوند تا یک کپیاز بسته ی فعلی ایجاد شود و به تاریخچه ی بسته ها اضافه شود. این عبارات در رفتار شبکه تاثیری ندارند اما امکاناستدلال در مورد تمامی تغییرات ایجاد شده در حین جا به جایی بسته در شبکه را فراهم می سازند. به صورت دقیق،معنای هر عبارت۱۶ نت کت با استفاده از معادلات زیر تعریف می شود:

(۲ .۲)

(۳ .۲)

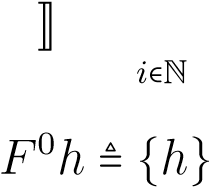
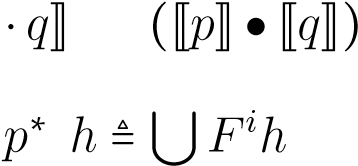
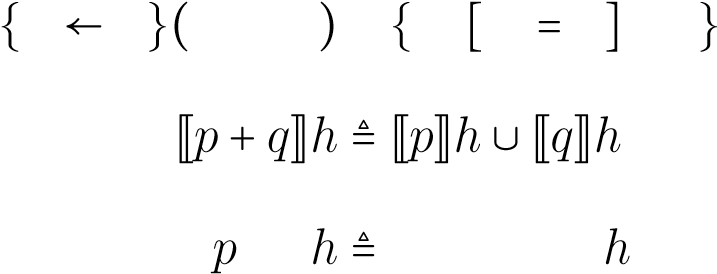
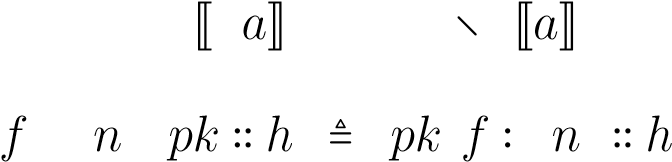
(۴ .۲)

(۵ .۲)

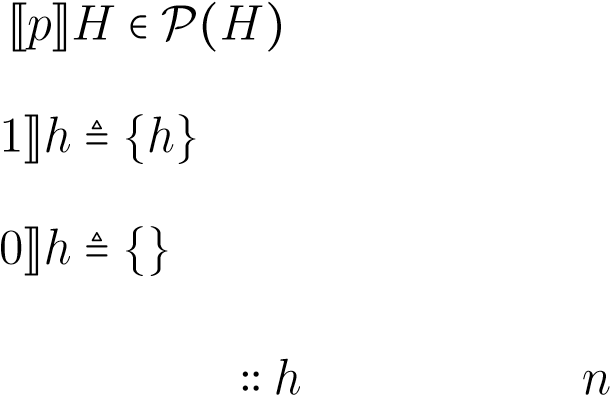
(۶ .۲)

(۷ .۲)

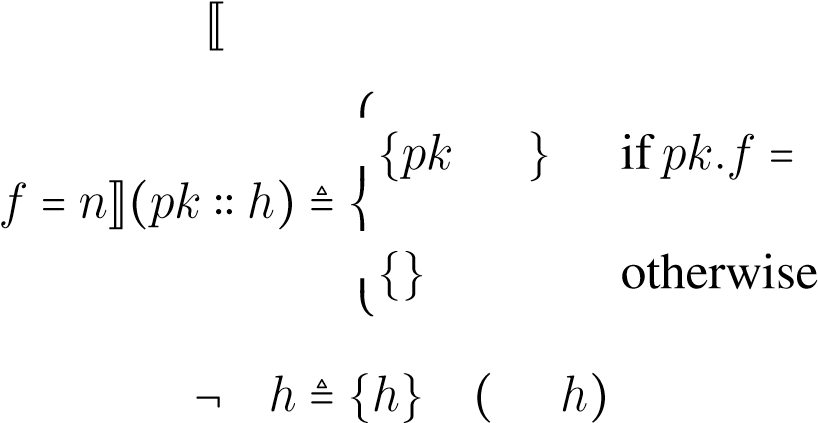
(۸ .۲)

(۹ .۲) J(۱۰ .۲)

J

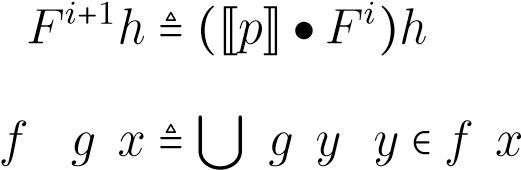


J



J

(۱۱ .۲)

 (۱۲ .۲) *pk*● *h*) *pk*{ ( *pk*)∣ *h* ( )} (۱۳ .۲)

J*dup*K( ∶ ) ≜ { ∶ ( ∶ )} (۱۴ .۲)

(

14Forward

15Drop

16Expression

#### ۱۰

در معادلات بالا فرض می شود که*H*  مجموعه ی تمامی تاریخچه های ممکن است. معادله ی ۲ .۲ بیان می کندکه معنی هر عبارت نت کت روی یک تاریخچه ی بسته یک مجموعه از تاریخچه ی بسته های حاصل از اعمال اینعبارت روی تاریخچه ی ورودی است. معادله ی ۳ .۲ بیان می کند که عبارت ۱ بسته را بدون شرط عبور می دهد.در مقابل معادله ی ۴ .۲ رها شدن بسته را با خروجی یک مجموعه ی خالی مدل می کند. معادله ی ۵ .۲ بسته ینخست ورودی را بررسی می کند و اگر مطابق با عبارت نبود بسته رها می شود. نقیض یک فیلتر در معادله ی ۶ .۲توصیف شده است. معادله ی ۷ .۲ مقدار*n*  را به فیلد*f*  بسته ی نخست تاریخچه اختصاص می دهد. معادله ی۸ .۲ جمع دو جمله را به صورت اجتماع تاریخچه های حاصل از اعمال هر یک از عملوند ها توصیف می کند.در معادله ی ۹ .۲ ترکیب متوالی دو جمله را به صورت ترکیبKleisli دو جمله که به شکل زیر تعریف می شودتوصیف می کند:

(*f* ● *g*)*x* ≜⋃{*g y* ∣ *y* ∈ *f x*}

در معادله ی ۱۰ .۲ حاصل اپراتور ستاره ی کلینی۱۷ معادل با اجتماع اعمال تابع های*Fi*  روی تاریخچه ی ورودیدر نظر گرفته شده است که تابع*Fi*  حاصل*i*  بار ترکیبKleisli عبارت*p*  است. در نهایت معادله ی ۱۴ .۲ یککپی از بسته ی نخست ورودی را به ابتدای خروجی اضافه می کند.

نت کت علاوه بر اصول موضوعه ی۱۸KAT اصول موضوعه ی زیر را هم شامل می شود تا دستگاه معادلاتی

17Kleene Star

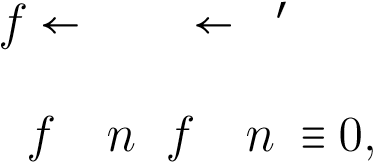
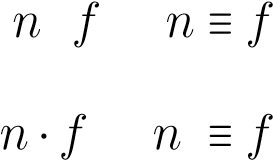
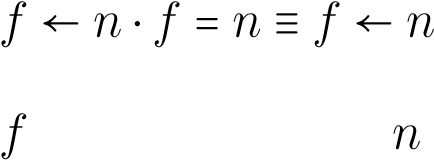
18Axiom

#### ۱۱

صحیح و کامل۱۹ داشته باشد:

*f* ← *n* ⋅ *ff*′ ← *nn*′ ≡ *ff*′ ←*nn*′ ⋅*ff* ←*n,n,* if*ff* ≠*ff*′ (۱۵ .۲)

*f* ← *n* ⋅ ′ = *n*′ ≡ *f*′ =*n* ′ *dup*⋅ ← if ≠ ′ (۱۶ .۲)

 *dup* ⋅ *f* = ≡ = ⋅ (۱۷ .۲)

(۱۸ .۲)

= ⋅ ← = (۱۹ .۲)

← *n*′ (۲۰ .۲)

= ⋅ *f*= *i*′ 1 if *n* ≠ *n*′ (۲۱ .۲) Σ*i* = ≡ (۲۲ .۲)

اصل های۱۵ .۲،۱۶ .۲،۱۷ .۲خواصجابه جایی۲۰عملیات هارابیانمی کنند. اصل۱۸ .۲بیانمی کندکهاختصاصمقدارn به یک فیلد و سپس فیلتر کردن بر روی این فیلد با همین مقدار معادل با عملیات اختصاص۲۱ به تنهاییاست. مشابه همین اصل برای یک فیلتر و سپس یک اختصاص هم در اصل ۱۹ .۲ مشخص شده. اصل ۲۰ .۲بیان می کند که در دنباله ای از اختصاص مقادیر به یک فیلد مشخص، تنها آخرین اختصاص تاثیر دارد. در اصل۲۱ .۲ مشخص شده است که مقدار یک فیلد نمی تواند دو مقدار متفاوت داشته باشد. در نهایت اصل ۲۲ .۲ بیانمی کند که عملیات مجموع فیلتر ها به ازای هر مقدار ممکن برای یک فیلد مشخص برابر عنصر همانی۲۲ است.

### **۳ .۳ .۲ توصیف رفتار شبکه با نت کت**

در ادامه فرض کنید که بخواهیم شبکه ای مانند شکل ۱ .۲ را توصیف کنیم که در آن بسته هایی که از نوعHSS هستند اجازه ی عبور نداشته باشند. همچنان می توانیم از سیاست توصیف شده توسط ۱ .۲ برای توصیف رفتارسوییچ ها استفاده کنیم. در ادامه می توان با اضافه کردن یک فیلتر به این عبارت، ویژگی دسترسی کنترل۲۳ را به این

19Sound and Complete

20Commutative

21Assignment

22Identity

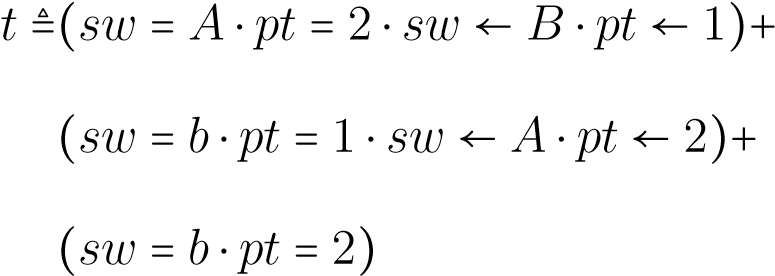
23Access Control

#### ۱۲

سیاست اضافه کرد تا همه ی بسته های از نوعSSH را رها کند:

*pAC* ≜ ¬(*typ* = *SSH*) ⋅ *p*

اما استفاده از عبارت بالا به تنهایی برای توصیف رفتار شبکه شکل ۱ .۲ کافی نیست. برای تکمیل این عبارت لازماست تا رفتار توپولوژی۲۴ شبکه هم به آن افزوده شود. در نت کت توپولوژی شبکه به عنوان یک گراف جهت داردر نظر گرفته می شود و رفتار آن در قالب اجتماع رفتار هر یک از پیوندهای۲۵ آن توصیف می شود. برای شبکه یشکل ۱ .۲ می توان از عبارت زیر برای توصیف توپولوژی شبکه استفاده کرد:



در نت کت در صورتی که سیاست و توپولوژی شبکه در قالب عبارت هایی توصیف شده باشند، رفتار کل شبکهدر واقع دنباله ای از اعمال این عبارت ها به صورت یکی در میان است. به عنوان مثال در شکل ۱ .۲ یک بسته ازمیزبان ۱ ابتدا توسط سوییچA پردازش شده، سپس روی لینک بین دو سوییچ جا به جا می شود و در نهایت توسطسوییچB پردازش می شود. در نت کت می توان این رفتار را به صورت*pAC* ⋅ *t* ⋅ *pAC*  توصیف کرد. با استفاده ازهمین شهود، رفتار کل شبکه را می توان در قالب عبارت زیر توصیف کرد:

(*pAC* ⋅ *t*)∗

در توصیف بالا فرض شده است که بسته ها می توانند به هر طریق ممکن وارد شبکه و از آن خارج شوند، اما اینرفتار همیشه مورد قبول نیست. به عنوان مثال در شبکه شکل ۱ .۲ اگر مکان های ورودی و خروجی شبکه را در

24Topology

25Link

#### ۱۳

قالب عبارت زیر توصیف کنیم:

*e* ≜ *sw* = *A* ⋅ *port* = 1 ∨ *sw* = *B* ⋅ *port* = 2

می توانیم رفتار انتها به انتهای۲۶ شبکه را به شکل زیر توصیف کنیم:

*pnet* ≜ *e* ⋅ (*pAC* ⋅ *t*)∗*e*

در حالت کلی تر، نیازی به توصیف ورودی و خروجی های شبکه در قالب یک عبارت نیست. پس اگر فرض شودکه مکان های ورودی شبکه توسط عبارت*in*  و مکان های خروجی شبکه در قالب عبارت*out*  توصیف شده باشند،رفتار یک شبکه در نت کت به صورت زیر تعریف می شود:

*in* ⋅ (*p* ⋅ *t*)∗ ⋅ *out*

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

### **۴ .۳ .۲ درستی سنجی برنامه های نت کت**

درستی سنجی یک شبکه و بررسی خواص آن در نت کت با استفاده از بررسی تساوی عبارت یک شبکه باعبارت های دیگر انجام می شود. به عنوان مثال در شبکه ی شکل ۱ .۲ برای بررسی اینکه همه ی بسته ها با نوع

SSH از میزبان ۱ رها می شوند کافی است تا تساوی زیر را ثابت کنیم:

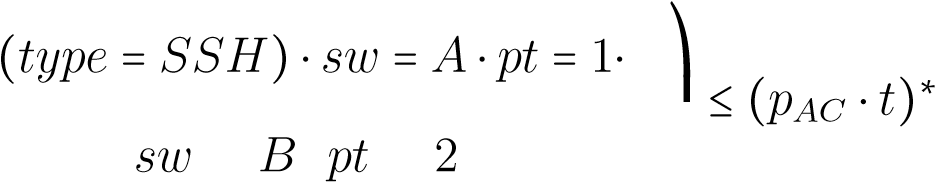
### *type sw A pt* 1

⎜⎛⎜⎜⎜⎜⎝⎜ = *SSHsw*(*p*=*ACB*⋅ ⋅⋅ *tpt*)=∗=⋅ 2⋅ = ⋅⎠⎞⎟⎟⎟⎟⎟⎟ ≡ 0

26End to End

#### ۱۴

از طرفی برای بررسی یک خاصیت در شبکه، مثلا امکان فرستاده شدن همه ی بسته هایی که از نوعSSH نیستند ازمیزبان ۱ به میزبان ۲*q* می ≡*p SSH p*+*q* توان به جای بررسی تساوی دو عبارت از نامساوی*p* ≤ *q*  استفاده کرد. این نامساوی کهخلاصه شده ی تساوی است بیان می کند که رفتار عبارت بخشی از رفتار عبارت*q*  است. بنابراین برایبررسی این مساله که شبکه ی شکل ۱ .۲ بسته های غیر از میزبان ۱ را عبور می دهد کافی است تا درستینامعادله ی زیر را بررسی کرد:

⎛⎜⎜⎝¬← ⋅ ← ⎟⎠

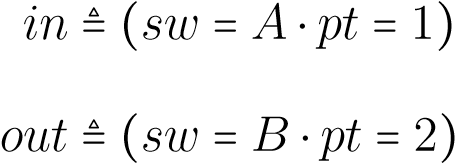
اصول موضوعه ی نت کت یک سیستم اثبات۲۷ را تشکیل می دهند که امکان اثبات این معادله یا نامعادله ها را فراهممی کنند. مثلا فرض کنید که سیاست دسترسی کنترل برای رها کردن بسته هایHSS ۱ .۲ را برای افزایش کاراییفقط در سوییچA انجام دهیم:

*pA* ≜ (*sw* = *A* ⋅ ¬(*typ* =SSH) ⋅ *p*) + (*sw* = *B* ⋅ *p*)

به طریق مشابه می توانیم این کار را در سوییچB هم انجام دهیم:

*pB*(*sw* = *A* ⋅ *p*) ≜ (*sw* = *B* ⋅ ¬(*typ* =SSH) ⋅ *p*)

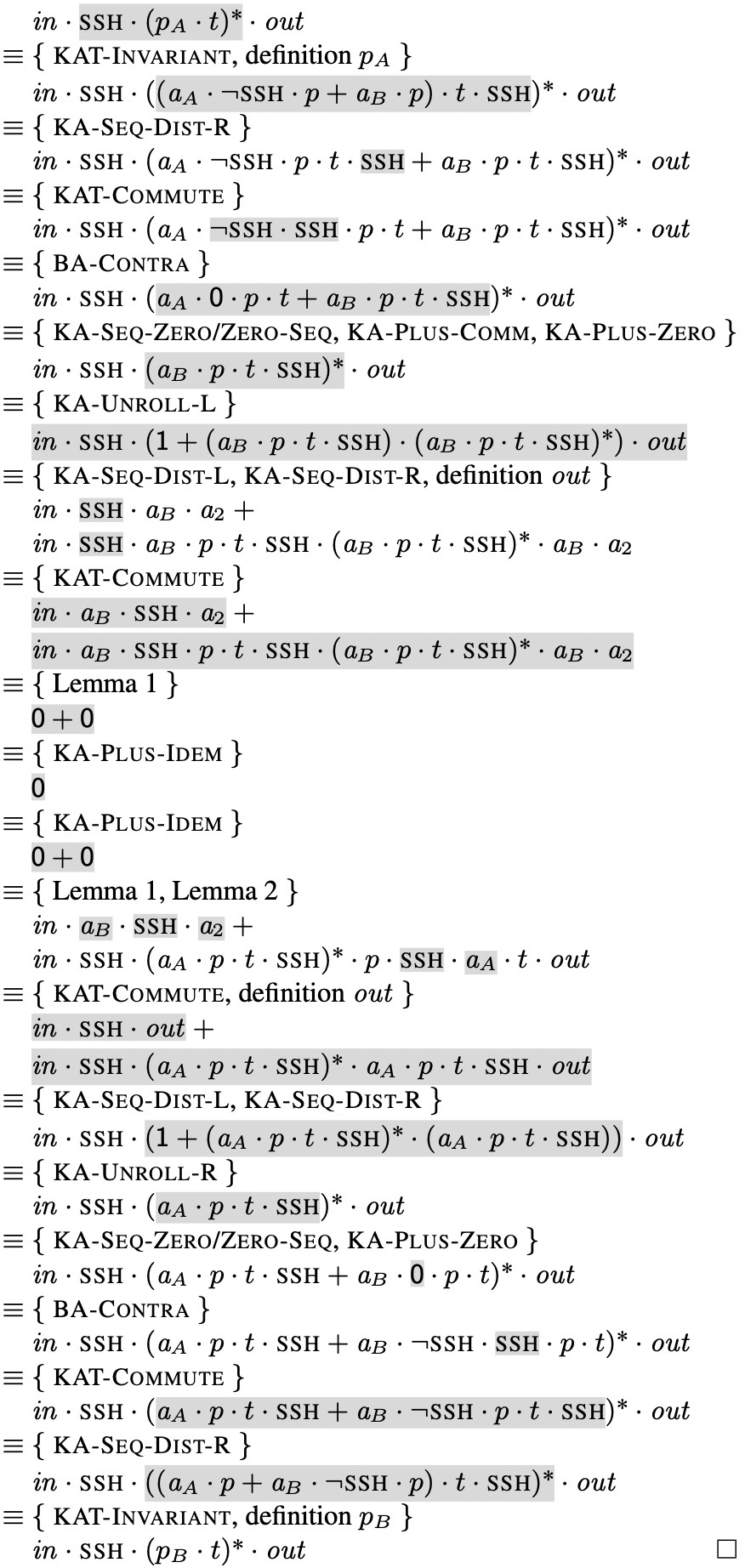
فرض کنید مکان های ورودی و خروجی به صورت زیر تعریف شده باشند:



اثبات معادل بودن دو سیاست*pA*  و*pB*  بر اساس این ورودی و خروجی در شکل ۲ .۲ ذکر شده است.

27Proof System

#### ۱۵



شکل ۲ .۲: اثبات معادل بودن*pA*  و*pB*  [۲]

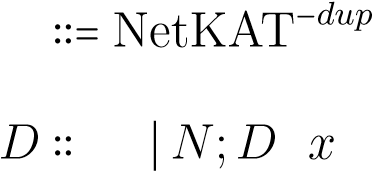
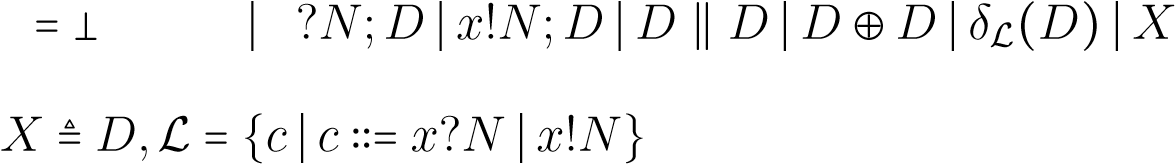
۱۶

## ۴ .۲ نت کت پویا

نک کت پویا۲۸ برای رفع برخی از کاستی های نت کت ارائه شده است [۷]. به صورت دقیق تر نت کت پویا،امکان توصیف به روز رسانی سیاست های شبکه و همچنین رفتار شبکه در مقابل چندین بسته را ممکن می سازد.

### **۱ .۴ .۲ دستور زبان نت کت پویا**

در نت کت پویا، از رفتار انتها به انتها۲۹ ی توصیف های شبکه در قالب عبارت های نت کت استفاده می شود.به این معنا که در نت کت پویا تنها خروجی حاصل از اعمال عبارات نت کت روی بسته ها اهمیت دارد و مسیریکه طی شده است در نظر گرفته نمی شود. به همین منظور دستور زبان نت کت پویا به صورت زیر تعریف می شود:

*N*

در سینتکس بالاNetKAT−*dup*  قسمتی از زبان نت کت است که عبارت های*dup*  از آن حذف شده است.عبارت های*dup*  در توصیف های نت کت تاثیری در رفتار یک عبارت ندارند و هدف از استفاده از آن ها ثبت یکاثر از هر بسته پس از پردازش توسط یکی از عناصر شبکه است و امکان استدلال بر روی رفتار شبکه را ممکنمی سازد. با توجه به این که در نت کت پویا رفتار انتها به انتهای یک عبارت نت کت مورد استفاده است، عبارت*dup* از دستور زبان کنار گذاشته شده است. نت کت پویا یک لیست از بسته های ورودی را پردازش می کند ویک لیست از مجموعه ی بسته های خروجی تولید می کند. اپراتور ترکیب متوالی۳۰*N*;*D*  باعث می شود که یکبسته از لیست بسته های ورودی توسط سیاست*N*  پردازش شود و سپس این بسته توسط عبارت*D*  پردازش شود.در نت کت پویا امکان ارتباط توسط عبارت هایی به شکل*x*!*N*  و*x*?*N*  توصیف می شوند که به ترتیب ارسال ودریافت یک عبارت نت کت را روی کانال*x*  توصیف می کنند. ترکیب موازی۳۱ دو عبارت توسط*D* ∥ *D*  توصیف

28DyNetKAT

29End to End

30Sequential Composition

31Parallel Composition

#### ۱۷

می شود. رفتار های غیرقطعی۳۲ توسط عبارت Lهایی به شکل*D* ⊕*D*  توصیف می شوند. عملگر*δ*L  باعث می شودتا از اجرای عملیات های غیر مجاز، که با مجموعه ی مشخص می شوند، جلوگیری شود.

### **۲ .۴ .۲ معنای عملیاتی نت کت پویا**

معنای عملیاتی۳۳ نت *H*کت پویا با استفاده از عبارت هایی به شکل (′*d,H,H*) تعریف می شوند که*d*  عبارت نتکت پویا فعلی است، لیست بسته هایی که در ادامه باید پردازش شوند و ′*H* لیست بسته هایی است که به= (*σ*( *i*صورت موفقیت *l γfi l*آمیز توسط شبکه پردازش شده ∶اند.*viF* → *σ* در اینجا فرض می شود که {*F* = {*f*1*,...,fn* ⟩⟨یک مجموعهاز فیلد های بسته ها است. یک بسته به شکل یک تابعN توصیف می شود. برای یک بسته مانند*σ*  تساوی*f vi* بیان می کند که مقدار فیلد در بسته ی برابر با است. یک لیست خالی از بسته ها با نمایشداده می شود. اگر یک لیست از بسته ها باشد*e*  لیستی است که حاصل از اضافه کردن بسته*σ*  به ابتدای لیست(*x,q*) به دست می*rcfg*آید. (*x,q*برچسب هر قانون که با)*rcfg*مشخص می *x*?*q x*!*q*شود به صورت یکی از شکل های*σ,σ*′)*,x*!*q,x*?*q* ) یا

تعریف می شود که به معنی انجام شدن و به صورت همگام۳۴ است. قوانین

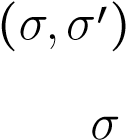
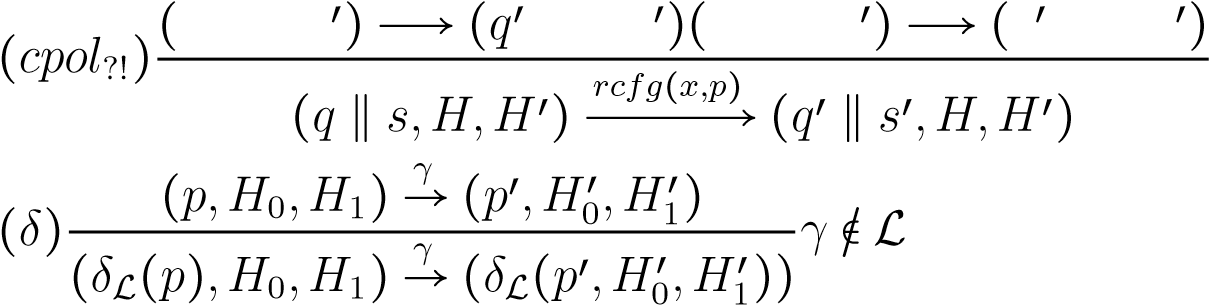
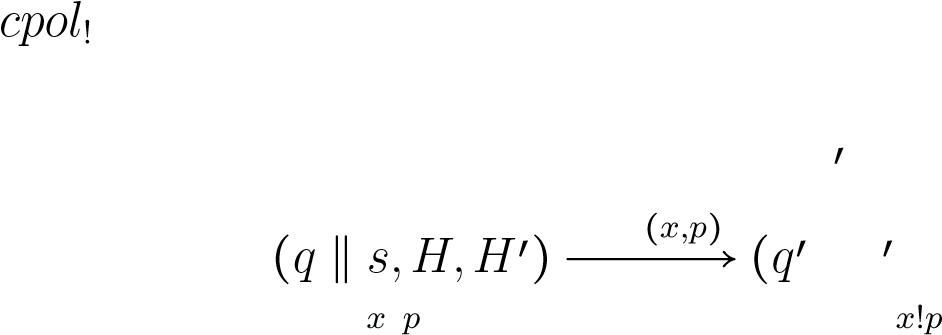
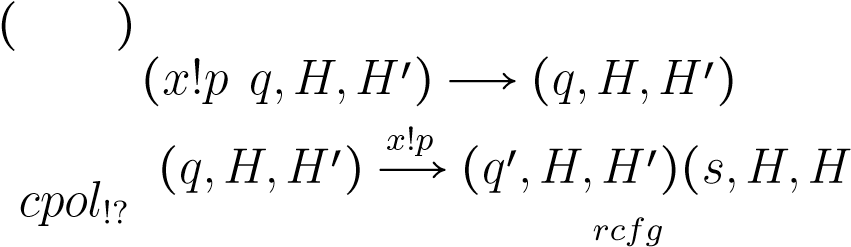
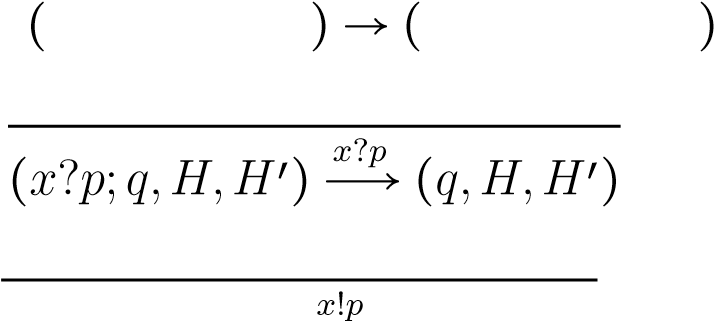
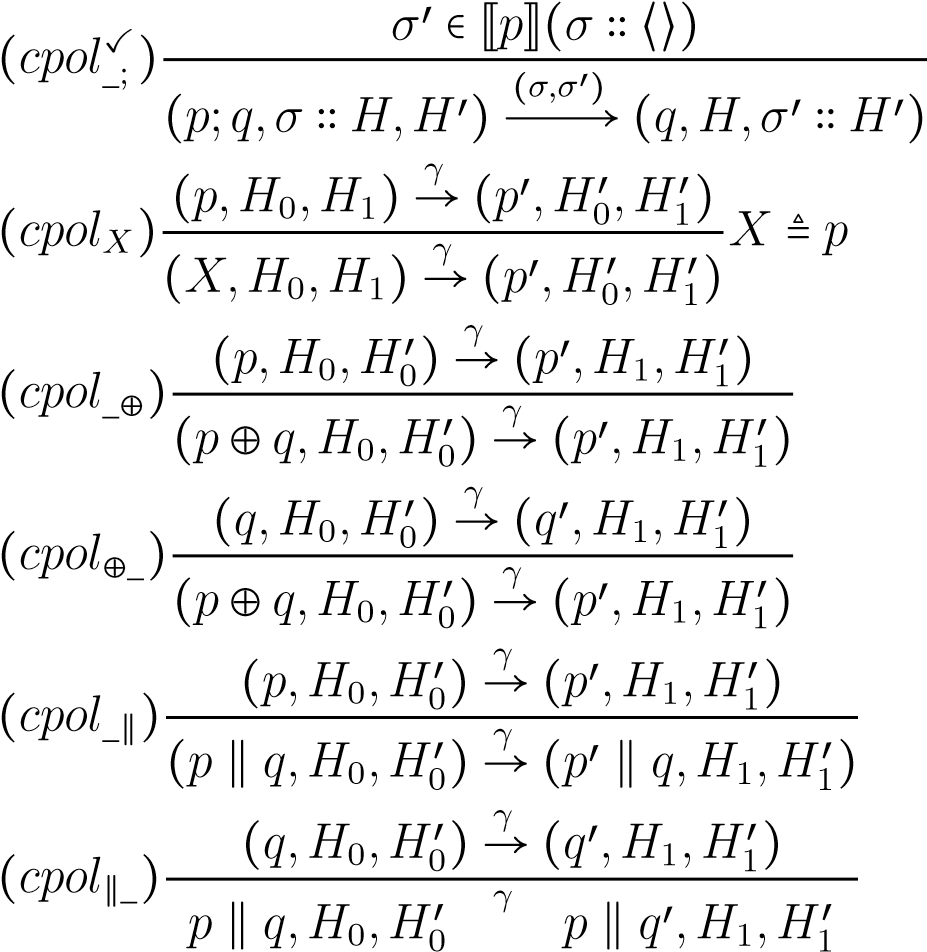
32Non-Deterministic

33Operational Seamntic

34Synchronized

#### ۱۸

زیر معنای عملیاتی نت کت پویا را تعریف می کنند:

(۲۳ .۲)

(۲۴ .۲)

(۲۵ .۲)

(۲۶ .۲)

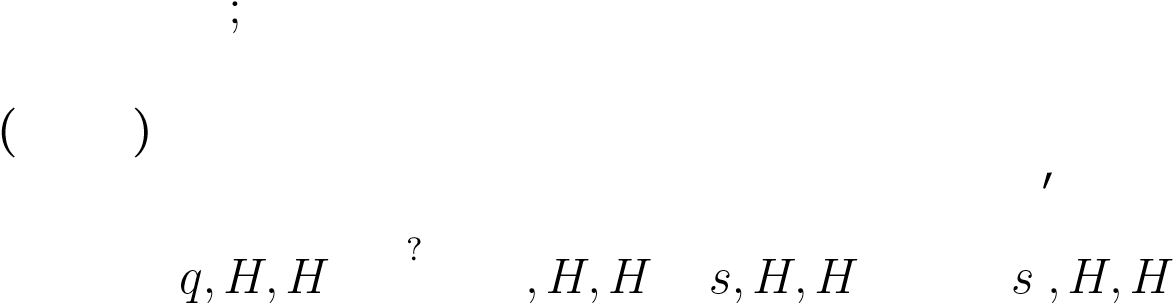
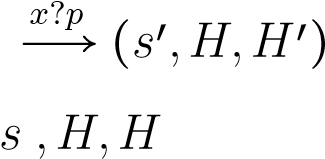
(۲۷ .۲)

(۲۸ .۲) *cpol*?) (۲۹ .۲)

(

(۳۰ .۲)

∥

) ) (۳۱ .۲)

(۳۲ .۲)

(۳۳ .۲)

قانون ۲۳ .۲ انجام یک عملیات مانندکه به معنای پردازش بسته ی ابتدایی لیست ورودی توسط عبارت*p* و افزودن خروجی حاصل از آن مانند ′ به لیست خروجی است را مشخص می کند. قانون ۲۴ .۲ بیان می کند کهرفتار متغیر*X*  که برابر با عبارت*p*  است معادل با رفتار عبارت*p*  است. قوانین ۲۵ .۲ و ۲۶ .۲ رفتار غیرقطعی راتوصیفمی کنندکهدرآنیکیازعملوند هاانتخابشدهوعملیات*γ*  راانجاممیدهد. قوانین۲۷ .۲و۲۸ .۲رفتاردو عبارت موازی را توصیف می کنند که در آن امکان اجرای عملیات توسط هر یک از عملوندها وجود دارد. قوانین

۲۹ .۲ و ۳۰ .۲ مشخص می کنند که ارسال یا دریافت پیام در نت کت پویا پردازشی روی بسته ها انجام نمی دهد.

#### ۱۹

*I*

*S*

*E*

*int*

*ext*

شکل ۳ .۲: مثال دیوار آتش

در نهایت همگامسازی۳۵ ارسال و دریافت پیام در پردازه های موازی توسط قوانین ۲۹ .۲ و ۳۰ .۲ توصیف شدهاست که در آن دو یکی از پردازه ها امکان ارسال و دیگری امکان دریافت پیام را دارد و در نتیجه در پردازه حاصل ازتوازی آن ها امکان انجام یک عملیات هنگام از نوع*rcfg*  وجود دارد که معادل همگام سازی عملیات های ارسالLو دریافت پیام است. قانون ۳۳ .۲*x*!*N x*?*N* محدود کردن پردازه توسط یک مجموعهL از عملیات های غیرمجاز را توصیفمی کند که در آن شامل عملیات هایی به فرم یا است.

### **۳ .۴ .۲ توصیف برنامه ها در نت کت پویا**

در ادامه چگونگی توصیف یک دیوار آتش۳۶ حالتدار۳۷ با استفاده از نت کت پویا بیان می شود. شبکه یشکل ۳ .۲ را در نظر بگیرید. در این شبکه هدف این است که امکان ارتباط از داخل شبکه به بیرون فراهم باشدولی امکان ارسال بسته از خارج شبکه ممکن نباشد. اما زمانی که یک بسته به خارج شبکه ارسال شد، دیوار آتشباید اجازه ی عبور بسته ها از بیرون را بدهد تا پاسخ بسته ها دریافت شوند. برای توصیف این شبکه می توان از

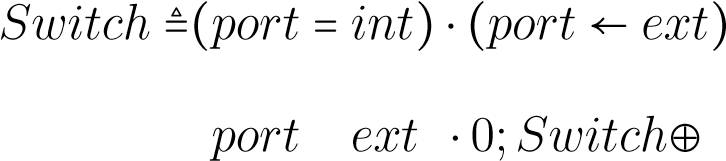
35Synchronization

36Firewall

37Stateful

#### ۲۰

عبارت نت کت پویای زیر استفاده کرد:

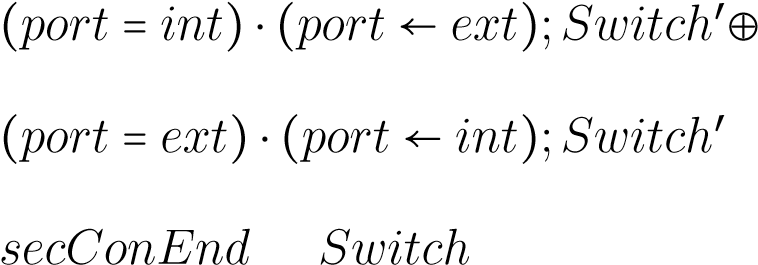
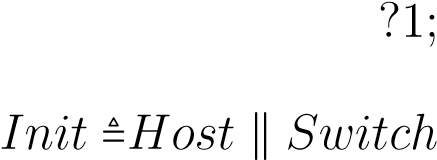
*Host* ≜*secConReq*!1;*Host* ⊕ *secConEnd*!1;*Host*

;*Switch*⊕

( = )

*secConReq*?1;*Switch*′

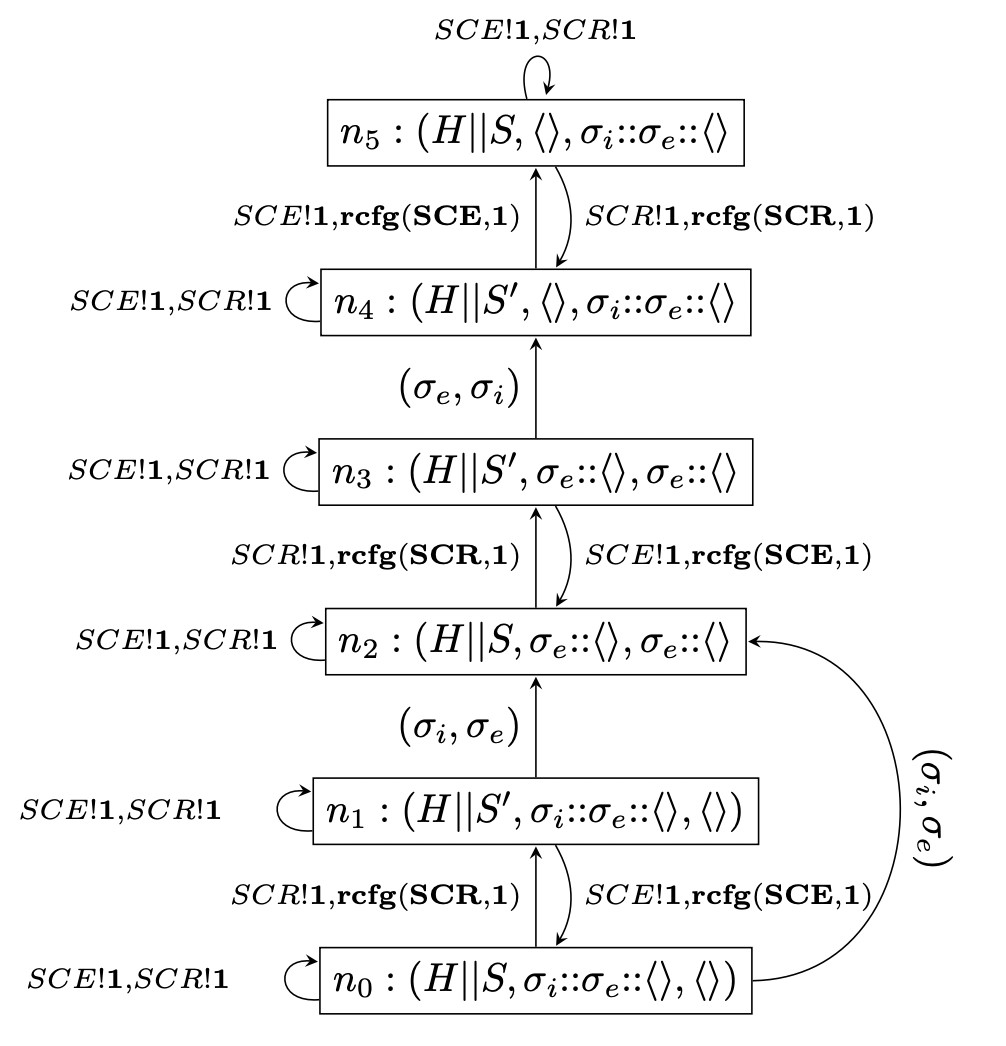
*Switch*′ ≜

  ⊕

در این توصیف عملیات*secConReq*  برای شروع ارتباط امن و*secConEnd*  برای خاتمه ی ارتباط امن در نظرگرفته شده است. بنابراین برنامه ی سوییچ پس از دریافت پیام برای شروع ارتباط امن تبدیل به برنامه ی ′*Switch*می شود که در آن اجازه ی ارسال بسته از پورت خارجی به پورت داخلی را می دهد. پس از دریافت پیام برایخاتمه ی ارتباط امن، برنامه به حالت اولیه ی خود بر می گردد و دوباره تمامی بسته های ورودی از پورت خارجیرا رها می کند. در نهایت رفتار کل شبکه با استفاده از ترکیب موازی یک میزبان و یک سوییچ در حالت اولیه خودتوصیف می شود. نمودار نمایش داده شده در شکل ۴ .۲ سیستم انتقال برچسب دار۳۸ این شبکه را در حالتی که یکبسته روی پورت ورودی و یک بسته روی پورت خروجی شبکه وجود دارد نشان می دهد. همانطور که در نمودارمشخص است، عملیات(*σe,σi*) که به معنای ارسال بسته از پورت ورودی به پورت خروجی است تنها در قسمتیاز این سیستم انتقال قابل دسترسی است که پیش از آن یکی از عملیات های 1?*SCR* یا (1*rcfg*(*SCR,* انجامشده باشند. بنابراین در این حالت شبکه تنها در صورتی که بسته خارجی را به داخل ارسال می کند که پیش از آنپیام آغاز ارتباط امن دریافت کرده باشد.

38Labeled Transition System

#### ۲۱



شکل ۴ .۲: سیستم انتقال برچسب دار برای شبکه ی دیوار آتش [۷]

## ۵ .۲ ساختمان رویداد

ساختمان رویداد۳۹ [۳۵] یک مدل محاسباتی۴۰ غیر برگ برگ شده۴۱ برای پردازه های هم روند۴۲ است. در اینمدل، برخلاف مدل های برگ برگ شده۴۳ مانند سیستم انتقال که هم روندی پردازه های موازی با انتخاب غیرقطعیمدل می شود، هم روندی پردازه ها به صورت صریح در مدل توصیف می شوند [۳۲].

**تعریف ۲. ۵. ۱.** ساختمان رویداد یک ساختمان رویداد یک سه تایی (⊢*,*#*E,*) است که در آن:

39Event Structure

40Computational Model

41Non-Interleaving

42Concurrent

43Interleaving

### ۲۲

۱.*E*  یک مجموعه از رویداد ها است

۲. # رابطه ی تعارض۴۴ ، یک رابطه ی دودویی متقارن و غیربازتابی بر روی مجموعهی*E*  است

۳.*Con* × *E*  ⊆⊢ رابطه ی فعال سازی۴۵ است که شرط زیر را برقرار می کند:

(*X* ⊢ *e*) ∧ (*X* ⊆ *Y* ∈ *Con*) ⇒ *Y* ⊢ *e*

در رابطه ی بالا*Con*  (مخفف،( Consistent زیرمجموعه ای از مجموعه ی توانی رویدادها است که اعضای آنفاقد تعارض باشند. به صورت دقیق تر داریم:

*Con* = {*X* ⊆ *E* ∣ ∀*e,e*′ ∈ *X.*¬(*e*#*e*′)}

**تعریف ۲. ۵. ۲.** به ازای هر ساختمان رویداد، می توانیم رابطه ی فعال سازی مینیمال را به صورت زیر تعریفکنیم:

1. ⊢*min e* ⇐⇒ *X* ⊢ *e* ∧ (∀*Y* ⊆ *X.Y* ⊢ *e* ⇒ *Y* = *X*)

همچنین در هر ساختمان رویدادی شرط زیر برقرار است:

1. ⊢ *e* ⇒ ∃*X* ⊆ *Y.X* ⊢*min e*

برای مشخص کردن وضعیت یک سیستم در هر لحظه از مفهومی به نام پیکر بندی۴۶ استفاده می شود و و یکمجموعه شامل رویدادهایی است که تا آن لحظه در سیستم رخ داده اند.

**تعریف***x* ⊆ *E* **۲. ۵. ۳.** اگر (⊢*,*#E = (*E,* یک ساختمان رویداد باشد، یک پیکربندی آن یک زیرمجموعه ازرویداد ها است که شرایط زیر را داشته باشد:

44Conflict

45Enabling

46Configuration

### ۲۳

∅

{

*r*

1

}

{

*r*

2

}

{

*r*

1

*,r*

2

}

{

*r*

1

*,r*

2

*,u*

}

شکل ۲. ۵

*s*

*h*

1

*h*

2

شکل ۲. ۶

*x* ∈ *Con* .۱

∀*e* ∈ *x.*∃*e*0*,...,en* ∈ *x.en* = *e* ∧ ∀*i* ≤ *n.*{*e*0*,...,ei*−1} ⊢ *ei* .۲

مجموعه ی همه ی پیکربندی های یک ساختمان رویداد مانندE  با (F(E نمایش داده می شود.

شبکه ی موجود در شکل ۶ .۲ را در نظر بگیرید. در این شبکه دو میزبان ۱ و ۲ به صورت هم روند یک بسته رابه سوییچ ارسال می کنند. این بسته ها شامل اطلاعات برای به روزرسانی مسیر های دیگر در شبکه هستند، بنابراینسوییچ پس از دریافت هر دوی این بسته ها آن ها را پردازش کرده و مسیر های خود را به روزرسانی می کند. فرضکنید رویدادهای 1*r* و 2*r* به ترتیب مشخص کننده دریافت یک بسته از میزبان ۱ و ۲ باشند و رویداد*u*  به روز رسانیسوییچ را مشخص کنند. برای مدل کردن این شبکه می توانیم از یک ساختمان رویداد به صورت زیر استفاده کنیم:

E = ({*r*1*,r*2*,u*}*,*∅*,*{(∅*,r*1)*,*(∅*,r*2)*,*({*r*1*,r*2}*,u*)})

یکی از روش های رسم نمودار برای ساختمان رویداد، رسم نمودار هسه۴۷ برای مجموعه ی پیکربندی های اینساختمان رویداد بر اساس رابطه ی زیرمجموعه است. برای مثالی که بیان شد می توان نموداری مطابق شکل ۵ .۲را رسم کرد.

47Hasse

۲۴

## ۶ .۲ مدل علّی

پیدا کردن تعریفی برای علت واقعی۴۸ مبحثی است که مورد مطالعه و تحقیق بسیاری قرار گرفته است. اینمساله به طور خاص در متون فلسفه مورد توجه قرار گرفته است. یکی از تعاریف علت واقعی که مورد توجهبسیاری قراری گرفته است، تعریفی مبتنی بر وابستگی خلاف واقع۴۹ است. مطابق این تعریف، رویداد الف علترویداد ب است اگر در شرایطی که رویداد الف اتفاق نیافته باشد، رویداد ب هم اتفاق نیافتند. در اینجا اتفاقنیفتادن رویداد الف خلاف واقع است، چون در سناریوی واقعی (سناریو ای که واقعا اتفاق افتاده و مشاهده شدهاست) رویداد الف اتفاق افتاده است و در نظر گرفتن شرایطی که در آن رویداد الف اتفاق نیفتاده باشد بر خلافواقعیت موجود است. اما این مدل به تنهایی امکان پیدا کردن علت مناسب را در همه ی موارد ندارد. به عنوانمثال سناریوی زیر را در نظر بگیرید که در آن سارا و بهرام هر کدام یک سنگ را برداشته و به سمت یک بطریشیشه ای پرتاب می کنند. در این سناریو، سنگ سارا زودتر از سنگ بهرام به بطری برخورد کرده و در نتیجه آن رامی شکند. در این سناریو واضح است که پرتاب سنگ توسط سارا علت شکسته شدن بطری است. فرض کنیدبخواهیم از علیت مبتنی بر خلاف واقع برای پیدا کردن این علت استفاده کنیم. بنابراین باید شرایطی را در نظربگیریم که سارا سنگ خود را پرتاب نکند. اما مشکل اینجاست که در این شرایط همچنان بطری شکسته می شود،چون اگر سارا سنگ خود را پرتاب نکند، بهرام همچنان سنگ خود را پرتاب می کند و در نتیجه این بار سنگ بهرامبه بطری برخورد کرده و آن را می شکند. بنابراین در این سناریو امکان تعریف پرتاب سنگ توسط سارا به عنوانعلت شکسته شدن بطری با استفاده از استدلال مبتنی بر خلاف واقع وجود ندارد. هالپرن۵۰ و پرل۵۱ برای حلکردن مشکلاتی از این دست، تعریف جدیدی از علت واقعی [۱۸] ارائه کردند. مدل ارائه شده توسط آن ها بهدلیل اینکه مبنای ریاضی دارد امکان استفاده از آن را در آنالیز و تحلیل سیستم های محاسباتی فراهم می کند. بههمین دلیل این تعریف در مقالات زیادی در حوزه ی دانش کامپیوتر مورد استفاده قرار گرفته است. برای تعریفعلت واقعی ابتدا برخی مفاهیم اولیه مورد استفاده در این تعریف توضیح داده می شوند. به صورت کلی فرضمی شود که دنیای مورد تحلیل توسط تعدادی متغیر تصادفی مدل شده است. اگر*X*  یک متغیر تصادفی باشد،یک رویداد به شکل*X* = *x*  تعریف می شود. برخی از این متغیر ها بر روی یکدیگر تاثیر گذارند. این وابستگی ها

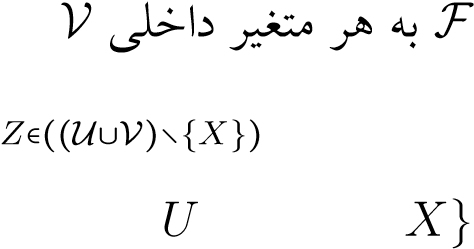
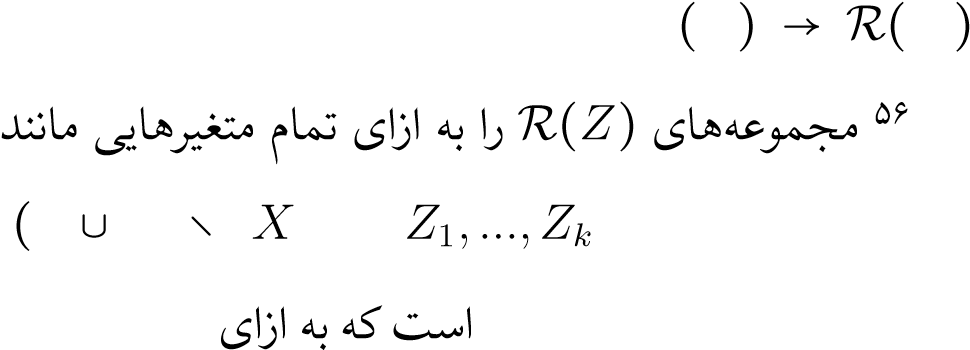
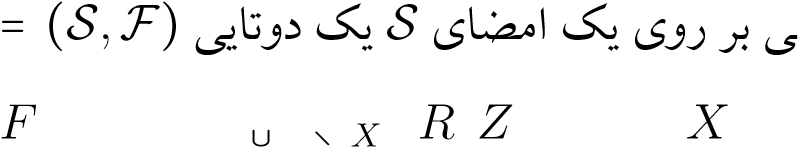
48Actual Cause

49Counterfactual

50Halpern

51Pearl

### ۲۵

در قالب مجموعه ای از معادلات ساختاری۵۲ مدل میشوند. هر یک از این معادلات در واقع یک مکانیزم یا قانونمشخص در این دنیا را مدل می کنند. متغیرها به دو دسته درونی۵۳ و برونی۵۴ تقسیم می شوند. متغیر های برونیمتغیر هایی در نظر گرفته می شوند که مقدار آن ها توسط عواملی که درون مدل نیستند تعیین می شوند. بنابرایندر یک مدل علی فرض می شود که مقدار این متغیر ها از قبل مشخص است. اما متغیر های درونی متغیرهاییهستند که مقدار آن ها بر اساس معادلات ساختاری تعیین می شود. به صورت دقیق تر، امضای۵۵ یک مدل یکسه تایی (S = (U*,*V*,*R است که در آنU مجموعه ی متغیرهای بیرونیV مجموعه ی متغیر های درونی وR دامنه ی مقادیر ممکن برای هر یک از متغیر ها را مشخص می کند. در این مدل فرض می شود که مجموعهی( )متغیر }V*X*) ∖R ∪ ( )) ∈های درونی محدود است×. مدل علّ*i Z i* =U 1*,...,k*M *X* ∶ ×*Z* است که در آن*X* یک تابع ({ } (U V))∈ اختصاص می دهد.نشانه گذاری ضرب خارجی در*Zi* ×*Z*مشخص می کند. (*k*بنابراین اگر فرض کنیم {V) { } = { (*z*1*,...,z* ، آنگاهU∪V)∖{ }) *Z*))∈ متشکل از چندتایی هایی به شکل هر*z* یک مقدار ممکن برای متغیر است. هر تابع، معادله ی یک متغیر را به ازای مقادیر تمام متغیرهای دیگر مشخصمی کند. به عنوان مثال اگر فرض کنیم*FX*(*Y,Z,U*) = *Y* + *U*  اگر داشته باشیم 2 =*Y* = 3*,U*  آنگاه مقدار*X* برابر ۵ خواهد شد. این معادلات امکان تفسیر آن ها بر اساس شرایط خلاف واقع را می دهند. به عنوان مثال درهمین مدل اگر فرض کنیم که*U* = *u*  می توانیم نتیجه بگیریم که اگر مقدار متغیر*Y*  برابر ۴ باشد آنگاه مستقل از( + =اینکه مقدار بقیه )[ ←ی متغیر ] ⊧ ( )ها در دنیای واقعی چه مقداری دارند، مقدار متغیر*X*  برابر 4+*u* خواهد بود که به صورت

4*M,u Y* 4 *X u*  نوشته می شود. توابع ذکر شده فقط برای متغیر های درونی تعریف می شوندو همانطور که پیش تر اشاره شد، برای متغیرهای بیرونی تابعی تعریف نمی شود و فرض می شود که مقدار آن ها ازقبل مشخص شده است [۱۷].

**مثال ۲. ۶. ۱.** یک جنگل را در نظر بگیرید که میتواند توسط رعد و برق یا یک کبریت رها شده دچار آتش سوزیشود. برای مدل کردن این سناریو از سه متغیر بولی۵۷ استفاده می کنیم:

52Structural Equations

53Endogenous

54Exogenous

55Signature

56Cross-Product

57Boolean

### ۲۶

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

در این مثال فرض می کنیم که مقادیر متغیر های برونی به گونه ای است که تمام شرایط لازم برای آتش سوزیجنگل در صورتی که رعد و برق اتفاق بیافتد یا کبریتی در جنگل رها شود را دارد (به عنوان مثال درختان جنگل بهاندازه ی کافی خشک هستند و اکسیژن کافی در هوا وجود دارد). در این مدل تابع متغیر*F*  را به گونه ای تعریف

(می U*,*V −V*X,*⃗∖R*X*⃗∣ *X*کنیم که داشته باشیم):*X,*⃗ U .*FF* (*u,L,M*⃗*M* )*x,*⃗=*u*⃗*L* ∨ *M* همانطور که پیش (F*,*تر بیان شد*MMX*=*x*(،Sاین مدل علّ*S*⃗*X* ی امکان بررسیمعادلات بر اساس شرایط خلاف واقع را می دهد. به صورت دقیق تر اگر یک مدل علی،*X*

⃗ ∖داخل*X*⃗ *FYL*Vاست. ⃗*x*یک بردار از متغیرهای درونی ویک زیرمدل*L*۵۸Falseاز*X*⃗ *M*تعریف می برداری از مقادیر متغیر هایبه صورت شهودی این مدل حاصل مداخله۵۹ =کنیم که در آن(⃗RF∣*X*V∖*Xx,*باشند مدل*x* =⃗(←S⃗*MX*محدود کردن*M*R*X*را با امضای*x*=به متغیر ⃗⃗*x*هایای در مدل است که در آن مقادیر را به متغیرهای اختصاص داده ایم. به صورت دقیق تر تعریف می کنیم ⃗←⃗ ⃗ ⃗←⃗ که ⃗←⃗*FYX*از تابع که در آن مقادیر را به متغیرهای اختصاص داده ایم به دست می آید. به عنوان مثال اگر*M*  مدلمثال ۱ .۶ .۲ باشد آنگاه در مدل ← معادله ی متغیر*F*  به*F M*  تبدیل می شود. این معادله دیگر بهمتغیر وابسته نیست بلکه با توجه به مقدار آن که در اینجا غلط است معادله ی جدیدی دارد. علاوه براین توجهکنید که در مدل*ML*←False دیگر معادله ای برای متغیر*L*  وجود ندارد. توجه کنید که در حالت کلی ممکن استیک بردار یکتا از مقادیر متغیر ها برای یک مدل وجود نداشته باشد که همزمان تمامی معادلات را حل کند. درمدل علّ ی یک بردار از مقادیر متغیرهای برونی ⃗*u* یک هم بافت۶۰ نامیده می شود. در مدل های بازگشتی به ازای یکهم بافت مشخص همیشه یک راه حل یکتا برای تمامی معادلات مدل وجود دارد. در ادامه فرض می شود که مدل هابازگشتی هستند. تعمیم مدل علّی برای مدل های غیربازگشتی در [۱۸] توضیح داده است. برای یک مدل می توانیک شبکه ی علّی ترسیم کرد. این شبکه یک گراف جهت دار است که به ازای هر متغیر یک گره در آن وجود دارد و

58Sub-Model

59Intervention

60Context

### ۲۷

یک یال بین دو گره وجود دارد اگر تابع متغیر دوم به متغیر اول وابسته باشد. به عنوان مثال شکل زیر شبکه ی علّیمثال ۱ .۶ .۲ را نشان می دهد:

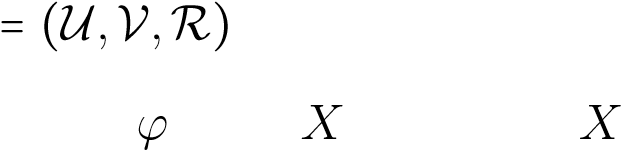
U

### L F

M

در ادامه برای سادگی رسم شبکه ی علی، متغیر های برونی را از آن ها حذف می کنیم.

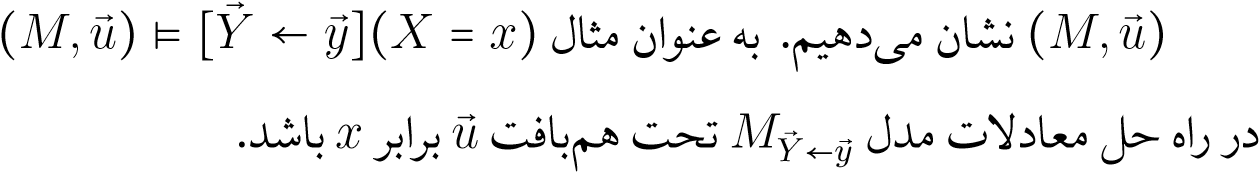
### **۱ .۶ .۲ علت واقعی**

در ادامه فرمول ( )R∈ های لازم برای تعریف علت واقعی توصیف میV∈شوند. اگر [S ← *k* 1 ← 1یک امضا باشدفرمول *X* = *x* یک رویداد بدوی۶۱ نامیده می شود که *,x* . فرمول*Y y ,...,Yk y* ]

ّیک فرمول علی پایه۶۲ نامیده می شود که در آن:

•*φ*  یک ترکیب بولی از رویداد های بدوی است

•*Y*1*,...,Yk*  متغیر های متمایز درV هستند• (*yi* ∈R(*Yi*

این فرمول به صورت خلاصه به شکل*Y*⃗ ← *y*⃗]*φ* ] نوشته می شود و اگر 0 =*k*  باشد آنگاه به صورت*φ*  نوشتهمی ⊧شود⃗.*Y*⃗ *y*به صورت شهودی یک فرمول به شکل*φY*⃗ ← *y*⃗]*φ* ] بیان می ⃗کند که در شرایط خّلاف واقع ای که در آنمقادیر به متغیر های اختصاص داده شده است فرمول برقرار است. یک فرمول علی به صورت یک ترکیببولی از فرمول ّهای علی پایه تعریف می شود. برقراری فرمول علی*ψ*  در مدل*M*  تحت هم بافت*u*  را به صورت

*ψ* برقرار است اگر مقدار متغیر*X*

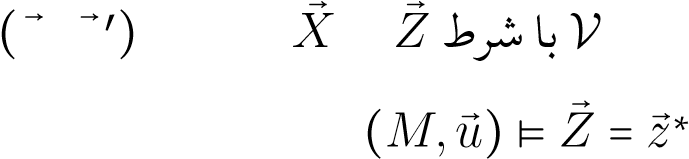
61Prime Event

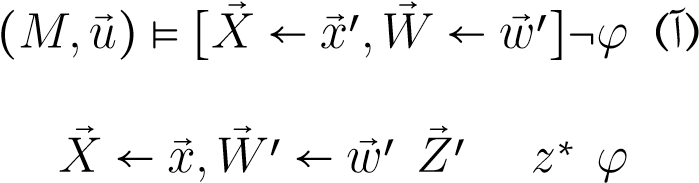
62Basic Causal Formula

#### ۲۸

**تعریف ۲. ۶. ۲.** فرمول ⃗*X*⃗ = *x* علت واقعی*φ*  ( که تاثیر۶۳ نامیده می شود) در (⃗*M,u*) است اگر شرایط زیربرای آن برقرار باشد:

(*M,u*⃗) ⊧ (*X*⃗ = *x*⃗) ∧ *φ* .۱

۲. ( ⃗ ⃗ )یک افراز مانند ( ⃗*Z,*⃗ *W*) از مجموعه ی متغیر های ⊆ و مقادیر*x,w*  برای متغیر های*X,W* وجود داشته باشد که داشته باشیم و شرایط زیر را برآورده کند:

∀*W*⃗ ′ ⊆ *W,*⃗ *Z*⃗′ ∈ *Z.*⃗ (*M,u*⃗) ⊧ [ *,* ← ⃗ ] (ب)

۳. ⃗*X* مینیمال باشد.

در این تعریف شرط اول بیان می کند که علت و تاثیر هر دو در شرایط واقعی برقرار هستند. شرط دوم بهدنبال پیدا کردن شرایطی است که تحت آن تاثیر به صورت غیر واقع به علت وابسته باشد. این شرایط متغیرهای⃗*W* و مقادیری مانند ′⃗*w* برای آن ها هستند. شرط ۲.آ بررسی می کند که تحت شرایطی که توسط ′⃗*W*⃗ ← *w* بهوجود می آید اگر علت مقداری متفاوت از مقدار خود در هم بافت واقعی داشته باشد اثر در مدل دیده نمی شود.شرط ۲.ب بررسی می کند که شرایط استفاده شده در ۲.آ عامل از بین رفتن اثر در ۲.آ نباشند. برای این منظور درشرایطی که علت مقدار واقعی خود را دارد در تمامی حالت هایی که متغیرهای شرایط می توانند داشته باشند بررسیمی شود که اثر همچنان برقرار باشد. شرط سوم در واقع بیان می کند که زیرمجموعه ای از علت وجود نداشته باشدکه همزمان شرایط ۱ و ۲ را برقرار کند. در تعریف بالا (′⃗*W,*⃗ *w*⃗′*,x*) یک شاهد۶۴ بر اینکه ⃗*X*⃗ = *x* علت*φ*  استتعریف می شود.

### **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 می کنیم.

برای مدل کردن این مساله متغیر های زیر را در نظر می گیریم:

63Effect

64Witness

#### ۲۹

*ST*

*SH*

*BS*

*BT BH*

شکل ۲. ۷

•*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

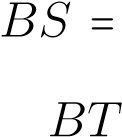
ابتدا فرض می کنیم که متغیر های*BT,ST*  تنها به متغیر های برونی وابسته اند. بطری در صورتی شکستهمی شود که هر یک از سنگ های سارا یا بهرام با آن برخورد کنند. بنابراین برای شکسته شدن بطری معادله ی*BS* = *BH* ∨ *SH* را در نظر می گیریم. نکته ی اصلی در این مساله این است که سنگ سارا زودتر از سنگبهرام به شیشه برخورد می کند، به همین دلیل لازم است تا این موضوع در مدل لحاظ شود. یک راه برای مدلکردن این مساله این است که معادله ی برخورد سنگ بهرام به شیشه را به گونه ای تعریف کنیم که تنها در صورتیکه سنگ سارا به بطری برخورد نکرده باشد آنگاه سنگ بهرام به بطری برخورد کند. بنابراین می توانیم معادله ی*ST BH* = *BT* ∧ ¬*SH* =را تعریف کنیم.*SH*علاوه بر این معادله ی برخورد سنگ سارا را بدون وابستگی بهّ برخوردسنگ بهرام تعریف می کنیم: . با توجه به این تعاریف برای معادلات می توانیم گراف علی شکل

۷ .۲False *BH* را برای این مدل رسم کنیم در این مدل می توانیم {False *W*⃗ *ST*= {*BSBT*= True =را به عنوان علت*BS* = True*w*  تعریف*ST*کنیم. برای برقراری شرط ۲ در تعریف علت واقعی شرایط و ′ را در نظر می گیریم.

در این شرایط چون مقدار برابر می شود، مقدار تنها وابسته به مقدار*SH*  و در نتیجه

می شود.=*W*⃗ = {*ST*}*,w* همچنین در این مدل*Z*⃗ = {*BH*} *BT* = True علت شکسته شدن شیشه نیست. مثلا فرض کنید که شرایطFalse ′ را در نظر بگیریم. در این شرایط اگر مقدار*BT*  را بهFalse تغیر دهیم مقدار*BS* هم غلط می شود. بنابراین شرط ۲.آ برقرار است. اما به ازای ′ شرط ۲.ب برقرار نمی شود. در این

#### ۳۰

حالت داریم:*M,u*⃗) ⊧ [*BT* ← True*BH,ST* ← False*,BH* ← *F*]False ) توجه کنید با وجوداینکه مقدار درست به اختصاص یافته اما چون مقدار به مقدار آن در هم بافت واقعی برگردانده می شوددر 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 برخلاف واقع را برطرف می کند و می تواند توضیح مناسبی در برخی از این مثال ها پیدا کند. نکته ای که باید به آنتوجه شود این است که هنوز روش یا معیاری برای این که چه تعریفی از علت واقعی تعریف مناسب است وجودندارد. تنها روش ممکن مقایسه تعاریف مختلف استفاده از آن ها در مساله ها و سناریوهای مختلف و بررسیتطابق علت به دست آمده با استفاده از این تعریف ها با شهود موجود از مساله است.

### **۳ .۶ .۲ مدل تعمیم یافته**

مدل علّی تعمیم یافته۶۵ یک سه تایی (S*,*F*,*E) است که (S*,*F) یک مدل علّی است وE*n* یک مجموعه از

مقداردهی های مجاز۶۶ برای متغیر های درونی است. به صورت دقیق تر اگر متغیر های درونی*X*1*,...,X*  باشندآن گاه*x*1*,...,xn*) ∈ E ) اگر*X*1 = *x*1*,...,Xn* = *xn*  یک مقداردهی مجاز است. یک مقداردهی دلخواه بهیک زیرمجموعه از متغیر های درونی مجاز است اگر امکان تعمیم به یک مقداردهی مجاز درE را داشته باشد.هدف از این تعریف جلوگیری از در نظر گرفتن علت هایی است که شرایط رخ دادن آن ها غیر محتمل است. باتوجه به تعریف مقداردهی مجاز، علت واقعی در یک مدل تعمیم یافته به گونه ای تعریف می شود که در شرط ۲فقط امکان مقداردهی های مجاز وجود داشته باشد. در [۱۸] تعریف دقیق علت واقعی در مدل تعمیم یافته بیاننشده است. در بخش بعدی تعریفی از علت واقعی در مدل تعمیم یافته ارائه می شود.

65Extended Causal Model

66Allowable Settings

۳۱

### **۴ .۶ .۲ علت واقعی بدون شرط**

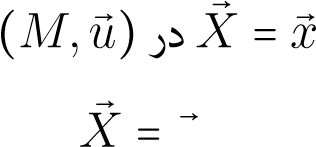
فرض کنید که⃗*X*⃗ = *x*⃗*W* یک علت واقعی برای*φ*  در (⃗*M,u*) با استفاده از شاهد (′⃗*x,*∅*,*∅) باشد. با توجهبه اینکه در اینجا یک بردار خالی است پس عملا شرط ۲. ب به بررسی شرط زیر تبدیل میشود:

∀*Z*⃗′ ∈ *Z.*⃗ (*M,u*⃗) ⊧ [*X*⃗ ← *x,*⃗ *Z*⃗′ ← *z*⃗∗]*φ*

با دقت در شرط بالا می توان دریافت که مقدار متغیر ها در فرمول های*X*⃗ ← *x,*⃗ *Z*⃗′ ← *z*⃗∗]*φ* ] با مقدار متغیر هادر هم بافت اولیه تفاوتی ندارد زیرا مقدار آن ها به مقداری که در هم بافت اولیه داشته اند برگردانده می شود. بنابرایندر شرط بالا می توان نتیجه گرفت:

(*M,u*⃗) ⊧ [*X*⃗ ← *x,*⃗ *Z*⃗′ ← *z*⃗∗]*φ* ⇐⇒ (*M,u*⃗) ⊧ *φ*

بنابراین شرط ۲.ب معادل با شرط ۱ می شود. با توجه به این موضوع می توان گزاره زیر را نتیجه گرفت:

**گزاره***φ* **۲. ۶. ۳.** *اگرx با شاهدی به شکلM,*∅*,*∅*u*⃗*,*)*x*⃗′)*φ*) *شرط های ۱،۲.آ و ۳ در تعریف ۲ .۶ .۲را برای برآورده کند آنگاه یک علت واقعی برای در* ) *است.*

۳۲

# فصل ۳مروری بر کا رهای پیشین

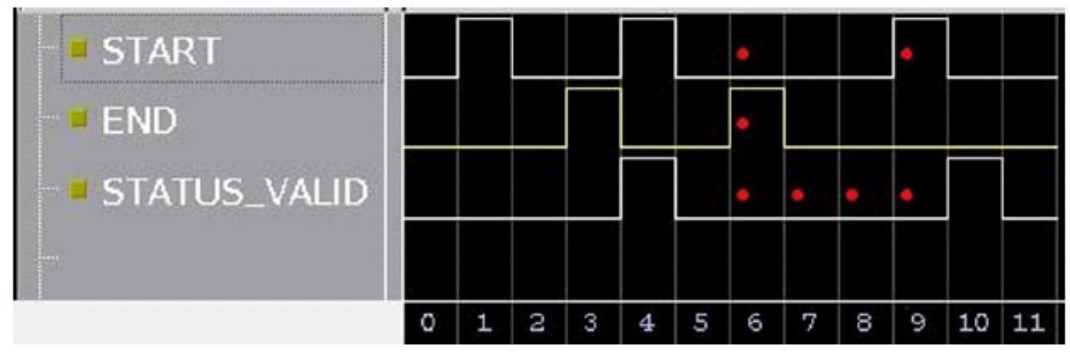
ابزار های درستی سنجی مشخص می کنند که آیا سیستم مطابق انتظار رفتار می کند یا خیر و در صورتی کهویژگی مورد انتظار توسط سیستم نقض شود این ابزارها می توانند مدرکی برای اثبات این مساله، مثل یک مثالنقض، تولید کنند. اما چنین مدارکی پاسخی به این سوال که چرا سیستم درست کار نمی کند نمی دهند. در نتیجهبرای دست یافتن به درک بهتری از اینکه چرا سیستم مطابق انتظار رفتار نمی کند طیف وسیعی از پژوهش ها برایپیدا کردن علت خطا انجام شده اند. استدلال مبتنی بر خلاف واقع که توسط لوئیس در [۲۶] ارائه شده است مبنایپیدا کردن علت پدیده ها در متون فلسفه است. هالپرن و پرل در [۱۸] فرمولاسیون ریاضی برای علت واقعی را ارائهکردند که مدلی برای به کارگیری استدلال مبتنی بر خلاف واقع برای پیدا کردن علت در سیستم های کامپیوتری رافراهم کرد. در ادامه پژوهش هایی در حوزه درستی سنجی که از تعریف هالپرن و پرل برای پیدا کردن علت خطااستفاده کرده اند را مورد بررسی قرار می دهیم.

## ۱ .۳ علت خطا در مثال نقض

در [۵] نویسندگان سیستم را به صورت یک سیستم انتقال۱ در نظر می گیرند که در آن هر حالت یک نگاشت ازیک مجموعه ی متغیر های بولی به مقادیر درست و غلط است. در این پژوهش با استفاده از تعریف علت واقعی در

1Transition System

### ۳۳



شکل ۱ .۳: رابط کاربری ابزارesaBeluREP

یک مثال نقض یک ویژگی توصیف شده درLTL ۲ یک دوتایی متغیر و حالت به عنوان علت واقعی در نظر گرفتهمی شود. در همین پژوهش یک الگوریتم تقریبی برای پیدا کردن همه ی علت ها در یک مثال نقض داده شده ارائهشده است و ابزاری برای نمایش این علت ها به صورت گرافیکی به کاربر توسعه داده شده و در ابزار درستی سنجیPE RuleBase متعلق بهIBM گنجانده شده است. برای مثال سیستمی را در نظر بگیرید که در آن زمانی کهتراکنش شروع شود سیگنالSTART  و زمانی که خاتمه یابد سیگنالEND  منتشر می شود. مدتی پس از اتمام یکتراکنش سیگنالSTATUS\_VALID  به معنی تایید تراکنش منتشر می شود. فرض کنید نیازمندی سیستم به گونه ایاست که تراکنش جدید نباید قبل از تایید تراکنش قبلی شروع شود. این نیازمندی را می توانیم در قالب ویژگیLTL زیر توصیف کنیم:

**G**((¬START ∧ ¬STATUS\_VALID ∧ END) → [¬START**U**STATUS\_VALID])

تصویر ۱ .۳ رابط کاربری ابزارPE RuleBase را پس از پیدا کردن یک مثال نقض برای این ویژگی نشان می دهد.در این تصویر نقاط قرمز علت های واقعی هستند که با الگوریتم تقریبی پیاده سازی شده پیدا شده اند. اینپژوهش یکی از کاربردی ترین استفاده ها از توضیح خطا و پیدا کردن علت خطا را نشان می دهد. در این پژوهشسعی شده است تا علت خطا در یک مثال نقض پیدا شود و به همین دلیل مقدار متغیر ها در حالت ها به عنوانعلت پیدا می شوند در حالی که در پژوهش جاری هدف پیدا کردن علت خطا در کل سیستم است و در واقعساختار های سیستم، مثلا وجود یا عدم وجود روابط تعارض یا فعال سازی به عنوان علت خطا پیدا می شوند. اما

2Linear Temporal Logic

۳۴

همانند پژوهش جاری در این پژوهش هم به شکل مستقیم و بدون تغییر از تعریفHP استفاده شده است.

## ۲ .۳ علت خطا در سیستم های قابل تنظیم

سیستم های قابل تنظیم۳ سیستم هایی هستند که با امکان افزودن یا کم کردن خصیصه ۴ های مختلف با تغییرتنظیمات۵ آن ها وجود دارد. رفع اشکال در این سیستم ها چالش بر انگیز است چون تعداد سیستم های ممکن باافزایش تعداد خصیصه ها به صورت نمایی زیاد می شود. پیدا کردن علت خطا در چنین سیستم هایی کمک می کندکه توسعه دهندگان صرفا برای رفع ایراد سیستم صرفا روی خصیصه ای تمرکز کنند که به عنوان علت خطا پیدا شدهاست و علاوه بر این به آن ها کمک می کند تا روش تنظیم مجدد۶ مناسب که منجر به خطا نشود را پیدا کنند. در[۳] وجود یا عدم وجود خصیصه ها در تنظیمات یک سیستم به عنوان متغیر ها در نظر گرفته شده است و مطابق باتعریف هالپرن و پرل وجود یا عدم وجود خصیصه ها به عنوان علت رخداد رفتار های قابل مشاهده در سیستم در نظر گرفته میشوند.

## ۳ .۳ علت خطا در پروتکل های امنیتی

در [۱۳] یک مدل برای توصیف برنامه های هم روند و پیدا کردن علت واقعی رخ دادن خطا در آن ها با استفادهاز تعریف هالپرن و پرل ارائه شده است. در این روش مجموعه ای از برنامه ها که با یکدیگر ارتباط دارند در نظرگرفته می شوند که اجرای منجر به خطای آن ها به شکل یک لاگ۷ ذخیره شده است. سپس یک زیرمجموعه ازعملیات های این برنامه ها به عنوان علت خطا در نظر گرفته می شود. این روش برای پیدا کردن علت خطا درپروتکل های امنیتی مورد بررسی قرار گرفته است و توانسته است ضعف هایی را در زیرساخت صدور گواهی۸ هایجاری بر اساس کلید عمومی۹ پیدا کند. در این پژوهش بر خلاف پژوهش جاری عملیات های سیستم به عنوان

3Configurable

4Feature

5Configuration

6Reconfiguraton

7Log

8Certification

9Public Key

۳۵

علت خطا در نظر گرفته می شوند.

## ۴ .۳ چک کردن علیت

در پژوهش [۲۵] نویسندگان تعریفی از علت واقعی که الهام گرفته از تعریفHP است ارائه می کنند والگوریتم آن ها بر اساس این تعریف در حین اجرای فرآیند وارسی مدل۱۰ علت ها را پیدا کرده و در نتیجه در انتهایوارسی مدل اگر سیستم ویژگی مورد نظر را نقض کرد به جای برگرداندن یک مثال نقض، رویداد هایی که علترخداد خطا بوده اند را بر می گرداند. در این پژوهش یک منطق برای توصیف یک دنباله از رویداد عملیاتهایسیستم ارائه شده است و فرمول های این منطق به عنوان علت خطا در نظر گرفته می شوند. این پژوهش هم همانند[۵] سعی بر پیدا کردن همه ی علت های بروز خطا دارد و علت ها عملا دنباله هایی از اجرای سیستم هستند. تفاوتاصلی این کار با پژوهش جاری در این است که در این پژوهش علت خطا در رفتارهای سیستم جستجو می شوددر حالی که در پژوهش جاری علت خطا در میان عناصر ساختاری سیستم جستجو می شود. این روش تنها برایویژگی های دسترس پذیری ارائه شده است. در [۶] نویسندگان این روش را برای ویژگی های دلخواه توصیف شدهتوسطLTL تعمیم دادند.

## ۵ .۳ علت واقعی در خودکاره های زمان دار

در [۲۲] نویسندگان از تعریفHP برای پیدا کردن علت خطا در خودکاره های زمان دار۱۱ استفاده کرده اند.در درستی سنجی خودکاره های زمان دار یک ابزار وارسی مدل بلا درنگ نقض ویژگی را در قالب یک رد تشخیصیزمان دار ۱۲ که در واقع یک مثال نقض است بر میگرداند. یکTDT در واقع یک دنباله متناوب از انتقال تاخیر۱۳و انتقال عملیات۱۴ ها است که در آن مقدار تاخیر ها به صورت سمبلیک مشخص شده اند. هدف این پژوهش پیداکردن مقادیری یا دامنه ای از مقادیر برای این تاخیر های سمبلیک است که بروز خطا را اجتناب ناپذیر می کنند یا

10Model Checking

11Timed Automata

12Timed Diagnostic Trace

13Transition Delay

14Delay Transition

### ۳۶

به عبارت دیگر علت واقعی هستند. در این پژوهش اما به صورت مستقیم از تعریفHP استفاده نشده است و براساس آن تعریفی برای علت واقعی نقض ویژگی در یکTDT بیان شده است.

## ۶ .۳ چارچوب علیت بر اساس رد سیستم

در [۱۶] نویسندگان این مساله را مطرح می کنند که تعریف ارائه شده توسط هالپرن و پرل ذاتا یک مدل براساس منطق گزاره ای۱۵ است و به همین دلیل برای درستی سنجی پردازه ها ایده آل نیست. در این پژوهش یکفرمالیسم و تعریف جدید برای علیت بر اساس تعریفHP ارائه می شود که در آن از رد ۱۶ های سیستم به جایمتغیر ها در مدلHP استفاده می شود و امکان ترکیب۱۷ چند مدل با یکدیگر را فراهم می کند.

## ۷ .۳ استدلال مبتنی بر علیت درLMH

در [۸] نویسندگان از مفهوم استدلال مبتنی در سیستم انتقال برچسبدار۱۸ وLMH ۱۹ [۲۰] استفاده کرده اند.در این پژوهش سیستم با استفاده از یک سیستم انتقال برچسب دار مدل می شود و رفتار ناامن توسط یک فرمولدر قالبHML توصیف می شود. سپس یک تعریف جدید که برگرفته شده از تعریفHP است با استفاده ازاین مدل ها برای علت واقعی بیان می شود. در این تعریف از مفهومی به نام عدم وقوع۲۰ رویدادها که پیش تر در[۲۵] مطرح شده بود استفاده می شود. شهود کلی مفهوم عدم وقوع در علیت این است که در کنار اینکه رخ دادنبرخی از رویداد ها منجر به خطا می شود، رخ ندادن رویداد ها هم می تواند به عنوان علت در نظر گرفته شود. درتعریف ارائه شده در این پژوهش مجموعه ای از محاسبه ۲۱ های سیستم به عنوان علت برقراری یک فرمولLMH در سیستم که رفتار نا امن۲۲ را توصیف می کند تعریف می شود. هر محاسبه شامل یک دنباله از عملیات هایسیستم در کنار تعدادی عملیات دیگر، که عدم وقوع آن ها هم جزئی از علت است، در نظر گرفته می شود. به

15Propositional Logic

16Trace

17Composition

18Labeled Transition System

19Hennesy Milner Loigc

20Non-Occurrence

21Computation

22Unsafe Behavior

### ۳۷

عبارت دیگر یک محاسبه را می توان شامل دو جز در نظر گرفت. جز اول یک اجرای سیستم است که منجر بهخطا می شود. جز دوم مجموعه ای از اجراهای سیستم است که منجر به خطا نمی شوند و حاصل برگ برگ شدن۲۳برخی از عملیات ها در جز اول این محاسبه هستند. عملیات های برگ برگ شده عملیات هایی هستند که عدم وقوعآن ها به عنوان علت بروز خطا در نظر گرفته می شود. در این تعریف علت واقعی به گونه ای تعریف شده است کهمحاسباتی که منجر به فعال شدن فرمولHML در سیستم می شوند به عنوان علت در نظر گرفته می شوند. دراین تعریف شروطی مشابه با شروط موجود در تعریفHP در نظر گرفته شده است. در [۹] نویسندگان تعریفخود را بهبود دادند تا تطابق بیشتری با تعریفHP داشته باشد. علاوه بر این در این پژوهش ثابت شده است کهاین تعریف از علت در سیستم هایی که ارتباط همگام۲۴ شده دارند قابل ترکیب نیست ولی در حالتی که سیستم هاارتباط همگام نداشته باشند امکان ترکیب یا شکستن آن وجود دارد. نتایج حاصل از این پژوهش یکی از انگیزه هایاصلی پژوهش جاری بود برای اینکه با انتخاب یک مدل معنایی یا تعریف علیت متفاوت امکان ترکیب آن برایسیستم های همگام شده بررسی شود. در ادامه به بررسی شباهت ها و تفاوت های این پژوهش و پژوهش جاریمی پردازیم اولا در این پژوهش تعریف جدیدی از علت واقعی ارائه شده است در حالی که در پژوهش جاریمستقیما از تعریف ارائه شده در [۱۸] استفاده شده است. در پژوهش جاری تمرکز بر پیدا کردن یک علت برایبروز خطا در سیستم است در حالی که در این پژوهش همه ی علل خطا مورد بررسی قرار میگیرند. پژوهشجاری علل خطا را در ساختار های سیستم جستجو می کند در حالی که این پژوهش در میان رفتار های سیستم بهدنبال علل خطا می گردد.

## ۸ .۳ جمع بندی

همان طور که بررسی شد پژوهش های متعددی در زمینه ی توضیح خطا ارائه شده است که نشان از اهمیتاین مساله در فرآیند درستی سنجی و اشکال زدایی دارد. همچنین تعریفHP هم مورد توجه زیادی برای پیدا کردنعلت خطا قرار گرفته است. یکی از مهم ترین تمایز های پژوهش جاری با پژوهش های پیشین در المان هایی استکه در آن علت خطا پیدا می شود. همانطور که بررسی شد در تمامی پژوهش های پیشین در این زمینه علت خطادر میان رفتارهای سیستم جستجو می شود. اما در پژوهش جاری رویکردی متفاوت استفاده شده است و علت

23Interleaving

24Synchronized

### ۳۸

خطا در میان ساختار های سیستم، مثلا هم روند بودن یا نبودن پردازه، انجام می شود. مساله ی دیگری که باید به آناشاره شود این است که در پژوهش جاری همانند [۵، ۱۰] به شکل مستقیم و بدون تغییر از تعریفHP استفادهمی شود.

۳۹

# فصل ۴روش و راه حل پیشنهادی

## ۱ .۴ مقدمه

در این فصل روش پیدا کردن علت خطا در یک برنامه ی توصیف شده در نت کت پویا توضیح داده می شود.در بخش اول مدل معنایی عبارات نت کت پویا با در قالب ساختمان رویداد تعریف می شود. در بخش دوم یکمدل علّی برای توصیف ساختمان رویداد مطرح می شود. در نهایت در بخش سوم شامل استفاده از این چگونگیترکیب این دو روش برای توضیح خطا در یک برنامه نت کت پویا توضیح داده می شود..

## ۲ .۴ مدل معنایی عبارات نت کت پویا در قالب ساختمان رویداد

در این بخش ابتدا انواع ترکیب و محدود سازی ساختمان های رویداد تعریف می شود. سپس با استفاده از اینتعاریف یک مدل معنایی برای عبارات نت کت پویا ارائه می شود.

**تعریف ۴. ۲. ۱.** فرض کنید (⊢*,*#E = (*E,* یک ساختمان رویداد باشد. به ازای یک مجموعه ی*A* ⊆ *E* ،

### ۴۱

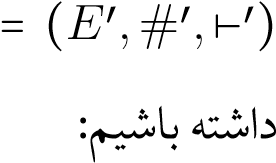
محدودیت۱E  به*A*  یک ساختمان رویداد به شکل زیر است:

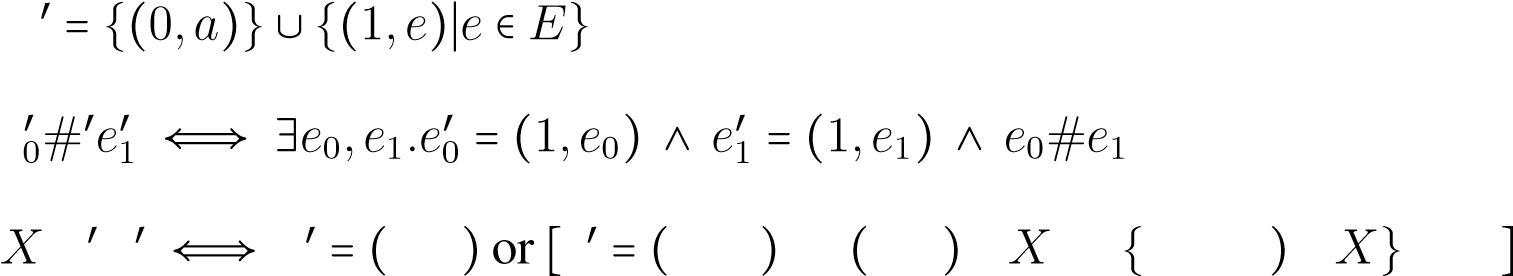
E⌈*A* = (*A,*#*A,*⊢*A*)

که اگر*ConA*  مجموعه ی تمامی زیرمجموعه های بدون تعارض*A*  درE⌈*A*  باشد آنگاه داشته باشیم:

*X* ⊆ *ConA* ⇐⇒ *X* ⊆ *A* ∧ *X* ∈ *Con*

*X* ⊢*A e* ⇐⇒ *X* ⊆ *A* ∧ *e* ∈ *A* ∧ *X* ⊢ *e*

**تعریف ۴. ۲. ۲.** فرض کنید (⊢*,*#*a* E = (*E,*یک ساختمان رویداد وE*a* یک رویداد باشد. ساختمان رویداد*a*E  که به معنای افزودن رویداد به عنوان پیشوند به است به گونه ای تعریف می شود که

 *E ,*

*e*

⊢ *e e* 0*,a e* 1*,e*1 ∧ 0*,a* ∈ ∧ *e*∣(1*,e* ∈ ⊢ *e*1

(⊢**تعریف** )**۴. ۲. ۳.***L* یک ساختمان رویداد برچسب دار۲ یک پنج تایی به شکل (*E,*#*,*⊢*,L,ll* ) است که در آن

*,*#*E,* یک ساختمان رویداد، یک مجموعه از برچسب ها (فاقد عنصر \* ۳ ) و یک تابع به فرم*l* ∶ *E* → *L* است که به هر رویداد یک برچسب اختصاص می دهد. یک ساختمان رویداد برچسب دار را به اختصار به صورت(E*,L,l*) نشان می دهیم.

**تعریف ۴. ۲. ۴.** در یک ساختمان رویداد رابطه ی ⩔ را به صورت زیر تعریف می کنیم:

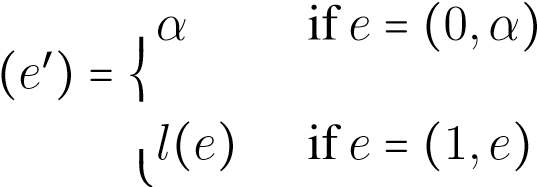
1Restriction *e* ⩔ *e*′ ⇐⇒ *e*#*e*′ ∨ *e* = *e*′

2Labeled Event Structure

۳ در ادامه از \* برای مشخص کردن رویداد های ناهمگام استفاده می کنیم. به همین دلیل این عنصر را به عنوان یک برچسب خاص ازمجموعه ی برچسب های ممکن کنار می گذاریم.

### ۴۲

**تعریف ۴. ۲. ۵.** فرض کنید (*α*E*,L ,l*) (E*,L,l*)یک ساختمان رویداد برچسب دار و*L α* یک برچسب باشد.∪({*α*(E=*,L,l*{*α*را به صورت یک ساختمان رویداد برچسب دار به شکل ′ تعریف می کنیم که در آن ′*L*و به ازای هر ′*e*′ ∈ *E* داشته باشیم:

⎧

*l*′

(**تعریف**⊢**۴. ۲. ۶.**) فرض کنید (0EE0 1= (*E*0*,*#0*,*⊢0*,L*0*,l*+0وE1 = (*E*1*,*#1*,*⊢1*,L*1*,l*1)E دو ساختمانرویدادبرچسب دارباشند. مجموعایندوساختمانرویداد رابهصورتیکساختمانرویدادبرچسب دار

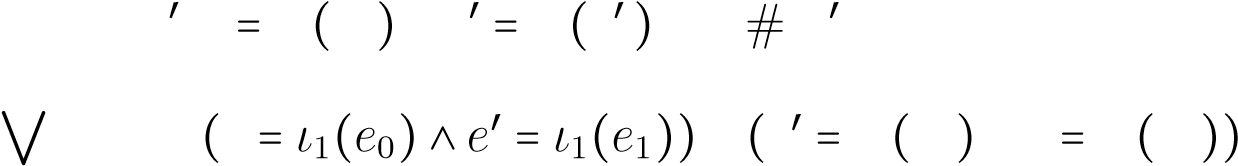
*E,*#*, ,L,l* تعریف می کنیم که در آن داشته باشیم:

*E* = {(0*,e*)∣*e* ∈ *E*0} ∪ {(1*,e*)∣*e* ∈ *E*1}

با استفاده از این مجموعه از رویداد ها توابع*ιk* ∶ *Ek* → *E*  را به ازای 1*,*0 =*k*  به شکل زیر تعریف می کنیم:

*ιk*(*e*) = (*k,e*)

به صورت شهودی در ساختمان رویداد جدید روابط تعارض قبلی بین رویداد ها حفظ می شود و علاوه بر آن بینهر دو رویدادی که یکی در 0*E* و دیگری در 1*E* بوده اند یک تعارض در نظر گرفته می شود. به صورت دقیق تررابطه ی تعارض در این ساختمان رویداد به شکل زیر تعریف می شود:

*e*#*e*′ ⇐⇒ ∃*e*0*,e*′0*.e* = *ι*0(*e*0) ∧ *e*′ = *ι*0(*e*′0) ∧ *e*0#0*e*′0

⋁

∃

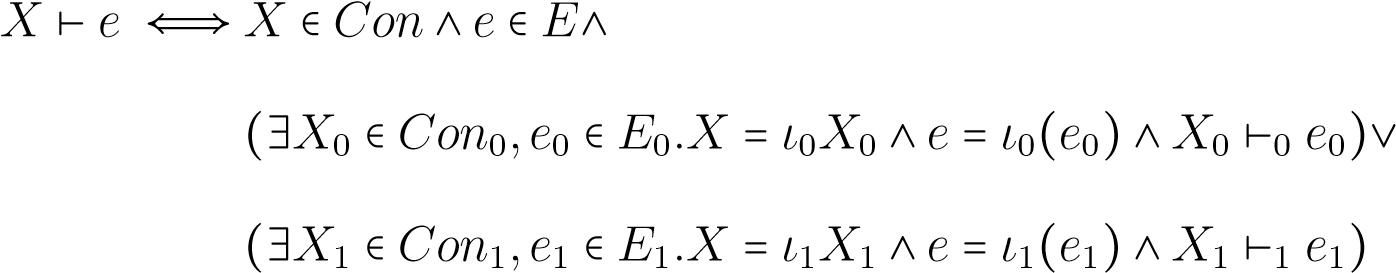
*e*1*,e*1*.e ι*1 *e*1 ∧ *e ι*1 *e*1 ∧ *e*1 1*e*1

∃*e*0*,e*1*. e* ∨ *e ι*1 *e*0 ∧ *e ι*1 *e*1

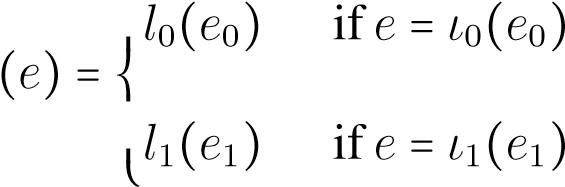
رابطه ی فعال سازی در ساختمان رویداد جدید عملا حاصل اجتماع روابط فعال سازی هر یک از عملوند ها است،

### ۴۳

به صورت دقیق تر داریم:



مجموعه ی برچسب ها را به صورت 1*L* = *L*0 ∪ *L* و تابع برچسب گذاری را به شکل تعریف می کنیم:

*l* ⎧

(**تعریف**E = (*E,*#**۴. ۲. ۷.***,*⊢*,L,l*فرض کنید که (0E0E=0×(E*E*10*,*#0*,*⊢0*,L*0*,l* و (1E1 = (*E*1*,*#1*,*⊢1*,L*1*,l* دو ساختاررویداد برچسب گذاری شده باشند. حاصلضرب آنها را به صورت یک ساختمان رویداد برچسب گذاریشده تعریف می کنیم که در آن رویداد ها به صورت زیر تعریف میشوند:

*E*0 ×∗ *E*1 = {(*e*0*,*∗)∣*e*0 ∈ *E*0} ∪ {(∗*,e*1)∣*e*1 ∈ *E*1} ∪ {(*e*0*,e*1)∣*e*0 ∈ *E*0 ∧ *e*1 ∈ *E*1}

*i*(*e*0*,e*1) = *ei*با توجه به این مجموعه *π* ∈رویداد ها توابعی به شکل*πi* ∶ *E* →∗ *Ei*  ⩔تعریف می کنیم که به ازای 1*,*0 =*i*  داشتهباشیم: . در اینجا رابطه ی تعارض را به کمک رابطه ی که پیش تر تعریف شد، به شکل زیر بهازای تمامی رویداد های*e,e*′ *E*  توصیف می کنیم:

*e* ⩔ *e*′ ⇐⇒ *π*0(*e*) ⩔0 *π*0(*e*′) ∨ *π*1(*e*) ⩔1 *π*1(*e*′)

### ۴۴

رابطه ی فعال سازی را به صورت زیر تعریف می کنیم:

*X* ⊢ *e* ⇐⇒ *X* ∈ *Con* ∧ *e* ∈E∧

### (*π*0(*e*) defined ⇒ *π*0*X* ⊢0 *π*0(*e*)) ∧ (*π*1(*e*) defined ⇒ *π*1*X* ⊢1 *π*1(*e*))

مجموعه ی برچسب های حاصلضرب را به صورت زیر تعریف می کنیم:

*L*0 ×∗ *L*1 = {(*α*0*,*∗)∣*α*0 ∈ *L*0} ∪ {(∗*,α*1)∣*α*1 ∈ *L*1} ∪ {(*α*0*,α*1)∣*α*0 ∈ *L*0 ∧ *α*1 ∈ *L*1}

در انتها تابع برچسب گذاری را به صورت زیر تعریف می کنیم:

*l*(*e*) = (*l*0(*π*0(*e*)*,l*1(*π*1(*e*))))

**تعریف ۴. ۲. ۸.***L* فرض کنید که (E Λ E = (*E,*Λ#*,*⊢E*,L,l*یک ساختمان رویداد برچسب دار باشد. فرض کنیدΛ

(*,l* یک زیرمجموعه ازΛ ∩باشد*,L*. ⊢*,* #*,*محدودیت*E*) بهΛ} *l*  ∈را به صورت(*E ,*# *,*⊢ ) = (*E,*#*,*⊢)⌈{*e* ∈ *E*∣*l*⌈(*e*Λ)و به شکل یک ساختمان رویداد برچسب گذاری

شده به شکل ′ ′ ′ ′ که در آن ′ ′ ′ است و

تابع برچسب گذاری معادل محدودسازی تابع به دامنه ی ∩*L*  است.

### **۱ .۲ .۴ معنای عبارات نت کت پویای نرمال**

در ادامه ابتدا فرم نرمال عبارات نت کت پویا را تعریف می کنیم. فرض کنید که فیلد های ممکن برای بسته ها

*α*←*π fα*1*,f*⋅*π*2*,...,f*← *k*باشندΣ. =یک فیلتر کامل۴ {*I* ∈به صورت*dup p I α* =*NetKATf*1 *q*= *n*A1*...f*=*dup*{*kα*=*i*⋅*nπpki*∣*i*و یک اختصاص کامل۵ به صورت*NetKATπ f*1 *n*1*...fk nk* تعریف می شود. می گوییم یک عبارت در*NetKAT* −*dup*  به فرم نرمال است اگربه شکلA ∈ ⋅ باشد که داشته باشیم . در عبارت قبل مدل زبانی −

می باشد. بر اساس لم ۵ در [۷] به ازای هر عبارت در − یک عبارت ′ به فرم نرمال وجود دارد

4Complete Test .*p* ≡ *p*′ :که داشته باشیم

5Complete Assignment

#### ۴۵

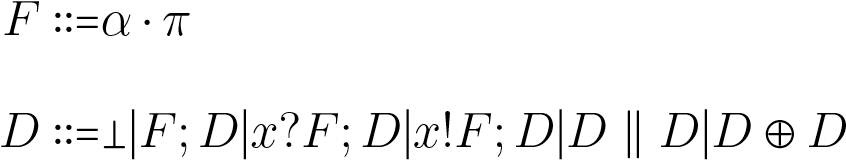
**تعریف ۴. ۲. ۹.** فرض کنید که*p*  یک عبارت نت کت پویا و*X*  متغیری باشد که در*p*  استفاده شده است. یکرخداد*X*  در*p*  محافظت شده ۶ است اگر و تنها اگر یکی از شروط زیر برقرار باشد:

•*p*  جمله ای به شکل*p*′;*t*  داشته باشد که در آن هیچ متغیری در ′*p* استفاده نشده باشد یا*X*  در*t*  رخ دادهباشد و رخداد تمامی متغیر های دیگر در ′*p* محافظت شده باشند.

• عبارت*p*  به فرم یکی از عبارت های*y*?*X*;*t*  یا*y*!*X*;*t*  باشد.

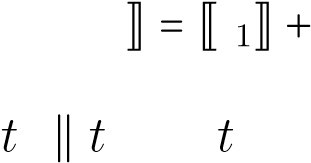
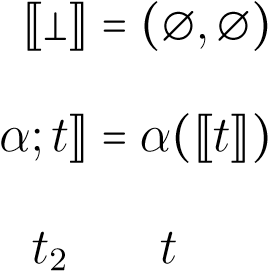
**تعریف ۴. ۲. ۰۱.** یک عبارت نت کت پویا مانند*p*  را محافظت شده می نامیم اگر همه ی رخداد های تمامیمتغیر ها در آن محافظت شده باشند.

**تعریف ۴. ۲. ۱۱.** زبان نت کت پویا نرمال را با دستور زبان زیر تعریف می کنیم:



با استفاده از لم ۵ در [۷] در لم ۹ همین مقاله ثابت شده است که به ازای هر عبارت محافظت شده*p*  درنت کت پویا یک عبارت معادل آن به فرم نرمال وجود دارد که داشته باشیم:*p* ≡ *q* . بنابراین در نهایت می توانیم هرعبارت محافظت شده نت کت پویا را به فرم یک عبارت نرمال با توجه به تعریف ۱۱ .۲ .۴ بنویسیم. در ادامه فرضکنیم کهA مجموعه ی الفبا شامل تمامی حروف به شکل*α*⋅*π,x*?*F,x*!*F*  باشد و داشته باشیم*α* ∈A*,L* ⊆A .

معنای عبارات نت کت پویای نرمال را با به صورت زیر تعریف می کنیم:

 JJ*t*1J⊕ =J K⌈A× ∖JJ*t*2KK

J

1 2 1 *t*2

*δL*(*t*)KK=J*t*K *L*

6Guarded

#### ۴۶

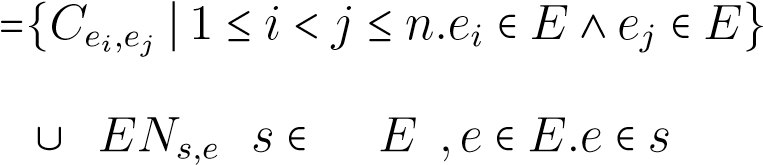
سمت چپ معادلات بالا عبارات نت کت پویای نرمال و در سمت راست ساختمان رویداد معادل هر یک مشخصشده است. در معادلات بالا (∅*,*∅) یک ساختمان رویداد که مجموعه ی رویداد ها و مجموعه ی برچسب هایآن تهی است را نشان می دهد.

## ۳ .۴ مدل علّی برای ساختمان رویداد

در ادامه نحوه ی توصیف یک ساختمان رویداد در قالب یک مدل علّی مطابق تعریفHP را بیان می کنیم.

(S F E)فرض کنیم که (⊢*,*#{S = (U*,*V*E,*R=){*e*1*,e*E2*,...,e*= (*E,n*یک ساختمان رویداد باشد. مدل علّ ∅ =U ی این ساختمان رویداد را به صورت =M

*, ,* تعریف می کنیم که در آن . در این مدل فرض می کنیم همه ی متغیر ها از نوع بولیهستند. همچنین در این مدل متغیر برونی در نظر نمی گیریم بنابراین داریم . اگر فرض کنیم مجموعه رویدادها به صورت باشد مجموعه ی متغیر های درونی را به صورت زیر تعریف می کنیم:

V

{*Ms,e s*∣ P*E*( *,e*) *E.e* /*s* } *PV*

∪ { ∣ ∈P( ) ∈ ∈/ } ∪ { }

به صورت شهودی به ازای هر عضو از رابطه های*min* ⊢*,*⊢*,*# یک متغیر درونی در نظر می گیریم که درست بودناین متغیر به معنای وجود عنصر منتاظر با آن در رابطه است.

به ازای (*x,y* ∈ P(*E* پوشیده شدن۷*x*  توسط*y*  را که با*x* ≺ *y*  نمایش می دهیم به صورت زیر تعریفمی کنیم:

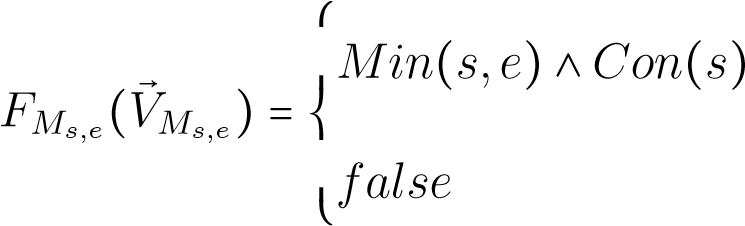
*x* ⊆ *y* ∧ *x* ≠ *y* ∧ (∀*z.x* ⊆ *z* ⊆ *y* ⇒ *x* = *z* or *y* = *z*)

همچنین به ازای هر متغیر*X* ∈V بردار*V*⃗*X*  را بردار شامل همه ی متغیر های درونی به غیر از*X*  تعریف می کنیم.

7Covering

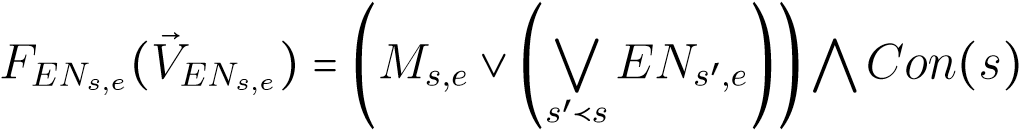
### ۴۷

با استفاده از این تعاریف توابع متغیر های درونی را به صورت زیر تعریف می کنیم:

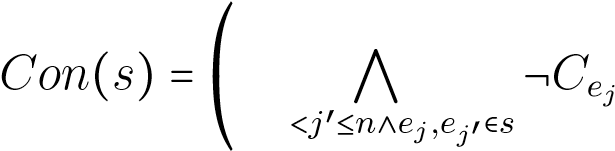
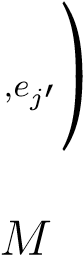
 *FCe,e* (*V*⃗*Ce,e* ) = ⎧⎪⎨⎪⎩*truefalse* if *e*#*e*′

### ′ ′ otherwise

if *s* ⊢*min e* otherwise



که در آن ها داریم:

1≤*j* 

*Min*(*s,e*) = ⎛⎝*s*′⊆*E.*(*s*′⊂*s*⋀∨*s*⊂*s*′)∧*e*∉*s*′ ¬ *s*′*,e*⎞⎠

فرض کنید کهE مجموعه ی تمامی سه تایی ها به فرم (′⊢*,*′#*E,*) باشد که داشته باشیم:

#′ ⊆ *E* × *E*

⊢′⊆P(*E*) × *E*

یک تابع به فرمE از مقدار فعلی متغیر→ (V}R(*V*  ّها در مدل عل*ES* ∶ ×*V* ∈V*V*∖{∈*PV*ی را به دست می ⃗*v*دهد. (تعریف می *v*⃗(*V*فرض کنیم ⃗*ES v*برداری شامل مقادیر متغیرهای {*PV* }∖کنیم که به صورت شهودی ساختمان رویداد حاصلVباشد. به ازای هر متغیر مانند مقدار آن در را با نمایش می دهیم. تابع را به گونه ای تعریف

### ۴۸

می کنیم که اگر (′⊢*,*′#*ES*(*v*⃗) = (*E,* آنگاه داشته باشیم:

∀*e,e*′ ∈ *E.eE ,e*#′*e*′*E.s*∧ *e*′#′*e* ⇐⇒ *v*⃗(*Ce,e*′) = True

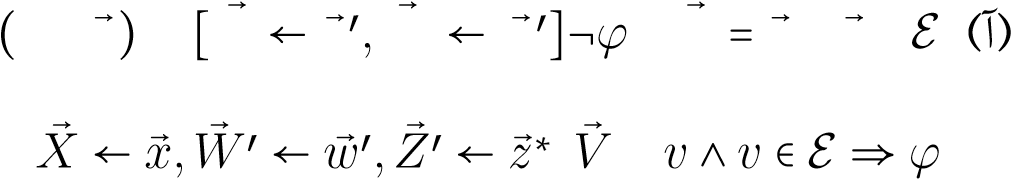
∀*s* ∈P( ) ∈ ⊢′ *e* ⇐⇒ *v*⃗(*ENs,e*) = True

در ادامه فرض می کنیم که رفتار نا امن۸ سیستمی که در قالب ساختمان رویداد مدل شده است، در قالب تابعمتغیر*PV*  توصیف شده است و در صورتی که رفتار نا امن در سیستم وجود داشته باشد مقدار آن درست و در غیر*ES*این صورت غلط است. در نهایت مقداردهی های مجازE را مجموعه ی مقداردهی هایی مانند ⃗*v* در نظر می گیریمکه خروجی تابع 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 ۲ .۶ .۲ برای مدل علّی تعمیم یافته است، استفاده می کنیم:

**تعریف**⃗*X*⃗ = *x***۴. ۳. ۱.** اگر (*MM,*=*u*⃗)(S*,*F*φ,*E)یک مدل علّی تعمیم یافته و ⃗*V* برداری از متغیر هایV ∖ *PV*  باشد،فرمول علت واقعی در است اگر شرایط زیر برای آن برقرار باشد:

(*M,u*⃗) ⊧ (*X*⃗ = *x*⃗) ∧ *φ* .۱

۲. ( ⃗*X,*⃗ *W*)یک افراز مانند ( ⃗*Z,*⃗ *W*) از مجموعه ی متغیر ⃗*z*های=*u*⃗) ⊧ *Z*⃗Vبا شرط ⃗*X*⃗ ⊆(*M,Z* و مقادیر (′⃗*x,*⃗ *w*) برای متغیر هایوجود داشته باشد که داشته باشیم ∗ و شرایط زیر را برآورده کند:

*M,u* ⊧ *X x W w* ∧ *V v* ∧ *v* ∈

∀*W*⃗ ′ ⊆ *W,*⃗ *Z*⃗′ ∈ *Z.*⃗ (*M,u*⃗) ⊧ [ ] = ⃗ ⃗ (ب)

۳. ⃗*X* مینیمال باشد.

در این تعریف برای اینکه تنها مقداردهی های مجاز برای پیدا کردن علت واقعی در نظر گرفته شوند شرط ۲تغییر یافته است. بند اضافه شده به شرط ۲.آ بیان می کند که تغییرات ایجاد شده در مدل یک مقدار دهی مجازباشند. بند اضافه شده در ۲.ب باعث می شود این شرط تنها در حالت هایی بررسی شود که مقداردهی داده شدهمجاز باشد.

8Unsafe Behavior

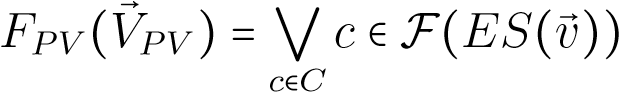
۴۹

## ۴ .۴ پیدا کردن علت خطا در نت کت پویا

با استفاده از تعاریف بخش های قبلی در این بخش به چگونگی پیدا کردن علت خطا در یک برنامه توصیفشده در نت کت پویا می پردازیم.

فرض می کنیم که یک عبارت نت کت پویا*p*  در اختیار داریم. ابتدا عبارت*p*  را به فرم نرمال مطابق تعریف۱۱ .۲ .۴ در می *q*آوریم. فرض کنیم عّبارتE *q*Mفرم نرمال عبارت*p*  باشد. اکنون فرض کنیمE = J*q*K ساختمانرویداد معادل باشد. اکنون مدل علی را بر اساس می سازیم وّ رفتار نا امن را در قالب تابع متغیر*PV*  اینمدل و به شکل یک شرط بر روی مجموعه ی پیکربندی های مدل علی توصیف می کنیم. در نهایت کافی استبرای پیدا کردن علت واقعی رفتار نا امن، علت واقعی*PV* = True  درM را بر اساس تعریف ۱ .۳ .۴ پیدا کنیم.توجه کنید که در اینجا محدودیتی برای چگونگی تعریف رفتار نا امن وجود ندارد و این تعریف می تواند هر شرطیبر روی مجموعه ی پیکر بندی های مدل علّی باشد.

مثلا اگر مجموعه ای از مثال های نقض سیستم در قالب پیکربندی های ساختمان رویداد وجود داشته باشد(مثلا مجموعه ی،( C میتوان رفتار نا امن را وجود یکی از این پیکربندی ها در سیستم توصیف کرد:



در مثالی دیگر اگر رفتار نا امن در قالب یک شرطunsafe روی برچسب های سیستم توصیف شده باشدمی توان رفتار نا امن را وجود یک پیکربندی که شامل یک رویداد که شرطunsafe را برآورده می کند توصیف کرد:

*FPV* (*V*⃗*PV* ) = ∃*c* ∈F(*ES*(*v*⃗))*.*∃*e* ∈ *c.*unsafe(*l*(*e*))

## ۵ .۴ بررسی چند ویژگی شبکه

در این فصل با استفاده از مدل علّی تعریف شده در بخش پیشین، علت نقض چند دسته از ویژگی های رایجدر شبکه را مورد بررسی قرار می دهیم.

در ادامه فرض می کنیم که فیلد*sw*  در همه ی توصیف های نت کت پویا وجود دارد. همچنین برای ساده تر

### ۵۰

*b*

*a*

*c*

*d*

شکل ۴. ۱

شدن توصیف ها از اصل زیر استفاده می کنیم:

*x* → *y* ≜ *sw* = *x* ⋅ *sw* ← *y*

## ۶ .۴ لیست سیاه

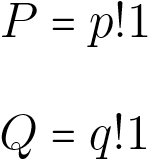
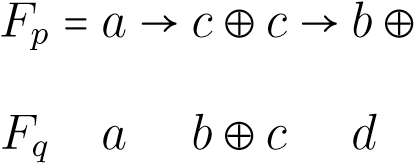
در این ویژگی، یک لیست سیاه۹ از مکان هایی در شبکه وجود دارد که نباید در شبکه به آن ها دسترسی وجودداشته باشد [۳۰]. مهم ترین استفاده از لیست سیاه را می توان برای اعمال سیاست های کنترل دسترسی در نظرگرفت که مثلا برخی از میزبان ها که دارای اطلاعات حیاتی هستند در لیست سیاه قرار میگیرند تا از بیرون بهآن ها دسترسی وجود نداشته باشد. به عنوان مثال دیگر ممکن است برخی از عناصر شبکه برای تعمیر برای مدتیکنار گذاشته شوند برای این منظور می توان آن ها را در لیست سیاه قرار داد تا دسترسی به آن ها سبب از دست رفتنبسته ها نشود.

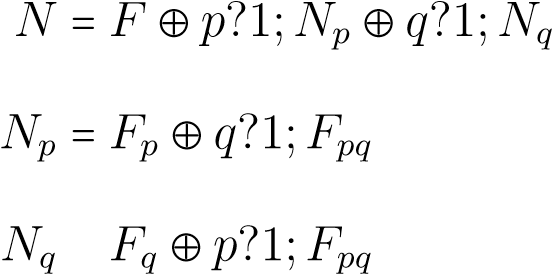
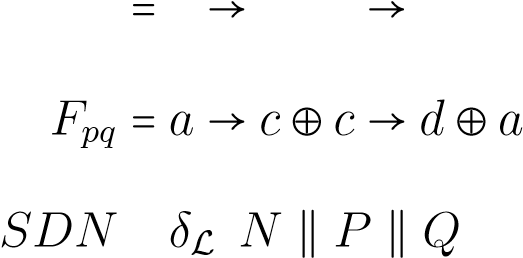
برای پیدا کردن علت نقض شدن ویژگی لیست سیاه شبکه ی رسم شده در شکل ۱ .۴ را در نظر بگیرید. دراین شبکه سوییچ*d*  در لیست سیاه قرار دارد، بنابراین در هیچ لحظه نباید از*a*  که ورودی شبکه است در دسترسباشد. بنابراین در شبکه عدم دسترسی*a*  به*d*  را به عنوان ویژگی در نظر می گیریم. در شبکه ی بالا ابتدا مسیر هاییکه با خط پررنگ مشخص شده اند وجود دارند. در ادامه هر یک از این مسیرها با مسیر های خط چین جایگزینمی شوند. فرض کنید به روز رسانی این مسیر ها توسط دو پردازه هم روند انجام می شود. واضح است که اگر هردوی این به روز رسانی ها انجام شوند دسترسی به سوییچی که در لیست سیاه قرار دارد ممکن می شود. اکنون فرض

9Blacklist

### ۵۱

کنید که از عبارات زیر برای توصیف این شبکه در نت کت پویا استفاده کنیم:

 *a* → *b*

 → *d*

== → ⊕ → L== {*p*(!1*,p*?1*,q*?1*,q*)?1}

*F a b c b*

در توصیف بالا پردازه های*P*  و*Q*  به ترتیب وظیفه ای ارسال پیام برای به روز رسانی مسیر های سبز و نارنجی رادارند. پردازهی*N*  رفتار ابتدایی شبکه و پردازه های*Np*  و*Nq*  به ترتیب رفتار شبکه را پس از به روز رسانی مسیر هایسبز و نارنجی توصیف می کنند. پردازه های*F,Fp,Fq,Fpq*  رفتارهای ارسالی۱۰ شبکه را توصیف می کنند. درنهایت رفتار کلی شبکه توسط پردازه ی*SDN*  توصیف شده است که حاصل ترکیب موازی پردازه های*N,P,Q* و جلوگیری از اجرای عملیات های همگام نشده است. در توصیف بالا امکان اجرای هر دو به روز رسانی وجوددارد بنابراین شبکه این امکان را دارد که به حالتی برسد که مسیری از*a*  به*d*  در آن وجود داشته باشد. برای مثالفرض کنید که*σa*  یک بسته وارد شده به شبکه باشد که داشته باشیم:*σa*(*sw*) = *a* . شکل ۲ .۴ بخشی از نمودارسیستم انتقال این شبکه را زمانی که این بسته به شبکه وارد شود نشان می دهد. اگر فرض کنیم*σd*  بسته ای باشد

که*Q P σd*(*sw*) = *d* همانطور که در نمودار مشخص است دو مسیر به حالتی که بسته از سوییچ*a*  به*d*  برسد وجوددارد. به دلیل هم روندی پردازه های و دو ترتیب برای اجرای این به روز رسانی ها وجود دارد و به همین دلیلدو مسیر منجر به خطا در این شبکه وجود دارد. اکنون می خواهیم علت بروز این خطا را پیدا کنیم. فرض کنید*PV* E =J*SDN*Kساختمان رویداد این شبکه وM مدل علیE  بر اساس مدل تعریف شده در ۳ .۴ باشد. در اینمدل تابع متغیر را به صورت زیر تعریف می کنیم:

*FPV* (*V*⃗*PV* ) = ∃*c* ∈F(*ES*(*v*⃗))*.*∃*e* ∈ *c.l*(*e*) = *a* → *d*

10Forwarding

### ۵۲

شکل ۴. ۲

*n*

0

∶

(

*SDN,σ*

*a*

*,*

⟨⟩)

*n*

1

∶

(

*δ*

L

(

*N*

*p*

∥

*Q*

)

*,σ*

*a*

*,*

⟨⟩)

*n*

3

∶

(

*F*

*pq*

*,σ*

*a*

*,*

⟨⟩)

*n*

4

∶

(

✓

*,*

⟨⟩

*,σ*

*d*

)

*n*

2

∶

(

*δ*

L

(

*N*

*q*

∥

*P*

)

*,σ*

*a*

*,*

⟨⟩

(

*rcfg*

(

*p,*

1

)

*rcfg*

(

*q,*

1

)

*rcfg*

(

*q,*

1

)

*rcfg*

(

*p,*

1

)

(

*σ*

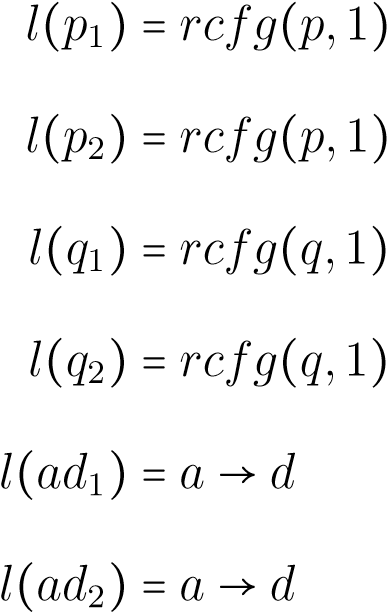
*a*

*,σ*

*d*

)

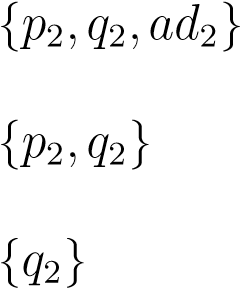
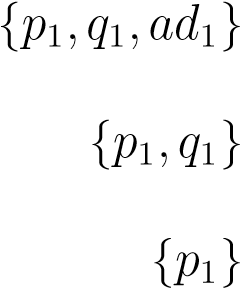
تابع بالا رفتار نا امن را وجود پیکربندی ای که شامل رویدادی با برچسب*a* → *d*  باشد توصیف می کند. با توجه*a* → *d*به ترتیب اجرای به روز رسانی ها در شبکه دو رویداد برای هر یک از عملیات های (1*rcfg*(*q,*1) ،*rcfg*(*p,*و در ساختمان رویداد وجود دارد. فرض کنید برای رویداد های مرتبط با این عملیات ها شش رویداد2*p*1*,p*2*,q*1*,q*2*,ad*1*,ad* وجود داشته باشد که برچسب آن ها به صورت زیر باشد:



شکل ۳ .۴ قسمتی از نمودار ساختمان رویداد این شبکه را نشان می دهد که در آن تمام پیکر بندی هایی

### ۵۳

شکل ۴. ۳



∅

که یکی از رویداد های 1*ad* یا 2*ad* را داشته باشند قابل دسترس باشد. با استفاده از مدل علّی در این مثال{ می (توانیم*q*1 *p*1 *PV C*(*p*2*,q*1*,q*2)1*,*)True= False{*,*Trueرا به عنوان یک علت برای نقض ویژگی لیست سیاه معرفی کنیم در صورتیکه از*C*(*p* ) به عنوان شاهد استفاده کنیم. ابتدا با توجه به شکل ۳ .۴ درE پیکربندی1*p*1*,q*1*,ad* قابلدسترسیاست. بنابراینمقدار صحیحاست. همچنینبینرویداد های و تعارضیوجود ندارد پس داریم*C*(*p*1*,q*1) = False . بنابراین شرط ۱ در تعریف ۱ .۳ .۴ برقرار است.

{1*p*1*,q*1*,ad*}اکنون فرض کنید که مقدار ({21*C,q*(2*p,ad*1*,q*12و*C*(*p*2*,q*2{)*pp* را برابر صحیح قرار دهیم. در این حالت هیچ یک ازپیکر بندیهای و 1*q*دیگر نمی 2*p*توانند عضوی از پیکربندی 2*q*های (⃗*ES*(*v* درM باشند زیرا با توجه به مقدار متغیرها بین روید اد های و و همچنین بین رویداد های و تعارض وجود دارد. پستحت این شرایط مقدار*PV*  غلط می شود بنابراین شرط ۲.آ هم برقرار می شود. برای بررسی برقراری شرط ۲.ب(⃗*v*)باید فرض کنیم که مقدار (*,q*2) *C*(*p*1*,q*1*ES*{غلط است12.*C*(*p*توجه کنید که در این شرایط پیکربندی {1*p*1*,q*1*,ad*} عضوی

از پیکربندی های است و مقدار روی این مساله تاثیری ندارد. همچنین برگرداندن مقادیربقیه متغیر ها به مقدار اولیه آن ها باعث حذف*p*1*,q*1*,ad* } از مجموعه ی پیکربندی ها نمی شود بنابراین شرط۲.ب هم برقرار است. با توجه به اینکه علت تنها شامل یک جمله است بنابراین شرط مینیمال بودن هم برقراراست. بنابراین در نهایت می توان نتیجه گرفت که (1*C*(*p*1*,q* یک علت واقعی برای بروز خطا در این شبکه است.در این مثال مشخص است که علت پیدا شده با علتی که به صورت شهودی باعث بروز خطا بوده است تطبیق

دارد.

### ۵۴

*a*

*b*

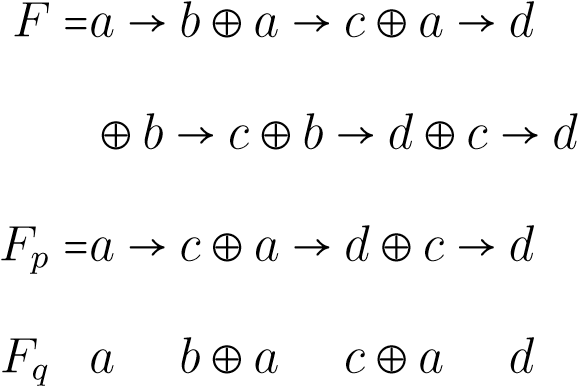
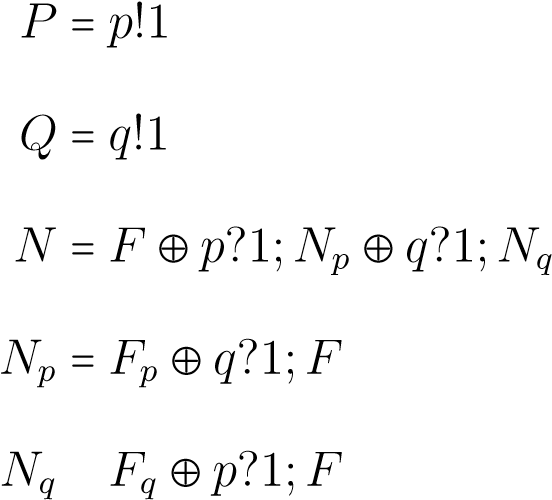
*c*

*d*

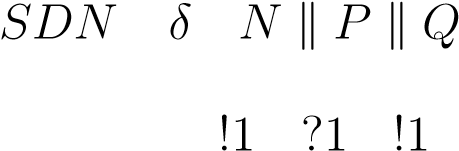
شکل ۴. ۴

## ۷ .۴ نبود دور

این ویژگی بیان می کند که شبکه نباید هرگز دارای دور باشد [۱۴]. وجود دور در شبکه می تواند باعثمشکلاتی مانند دور زدن یک بسته در شبکه بدون رسیدن به مقصد و در نتیجه کاهش کارایی شبکه شود. بهعنوان مثال شبکه ی رسم شده در شکل ۴ .۴ را در نظر بگیرید. در ابتدا مسیری از*a*  به*d*  وجود دارد. در این شبکهدو به روز رسانی بر روی سوییچ های*a*  و*c*  انجام می شود تا مسیر جدیدی از*a*  به*d*  ایجاد شود که اینبار ابتدا از*c* عبور می کند. می توانیم از توصیف نت کت پویای زیر برای توصیف این شبکه استفاده کنیم:

  = →*b c* →*b b* →*b d*

=

 L== {L*p*( *,p ,q ,q*)?1 ⊕⊕ *c* →→ *b* ⊕⊕ *c* →→ *c* ⊕⊕ *c* →→ *d*

}

در توصیف بالا پردازه های*P*  و*Q*  به ترتیب وظیفه ای ارسال پیام برای به روز رسانی مسیر های سبز و نارنجی رادارند. توجه کنید که در این توصیف پس از اجرای هر دو به روزرسانی رفتار ارسالی شبکه همانند رفتار اولیه خودمی شود. همانطور که در شکل ۴ .۴ مشخص است اگر به روز رسانی نارنجی پیش از به روز رسانی سبز انجامشود در شبکه یک دور شامل گره های*c*  و*b*  ایجاد می شود. شکل ۵ .۴ قسمتی از سیستم انتقال برچسب دار شبکهرا در حالتی که یک بسته ورودی روی سوییچ*b*  وجود داشته باشد را نشان می دهد. همانطور که در شکل مشخص

### ۵۵

شکل ۴. ۵۵۶

*n*

0

∶

(

*SDN,σ*

*b*

*,*

⟨⟩)

*n*

1

∶

(

*δ*

L

(

*N*

*p*

∥

*Q*

)

*,σ*

*a*

*,*

⟨⟩)

*n*

3

∶

(

✓

*,*

⟨⟩

*,σ*

*b*

)

*n*

2

∶

(

*δ*

L

(

*N*

*q*

∥

*P*

)

*,σ*

*a*

*,*

⟨⟩)

*n*

4

∶

(

*F,σ*

*b*

*,*

⟨)⟩

*n*

5

∶

(

✓

*,*

⟨⟩

*,σ*

*d*

)

*rcfg*

(

*p,*

1

)

*rcfg*

(

*q,*

1

)

(

*σ*

*b*

*,σ*

*b*

)

*rcfg*

(

*q,*

1

)

*rcfg*

(

*p,*

1

)

(

*σ*

*b*

*,σ*

*d*

)

{

*p*1*,q*1} { } { } { }

∅

{

*p*

1

}

{

*q*

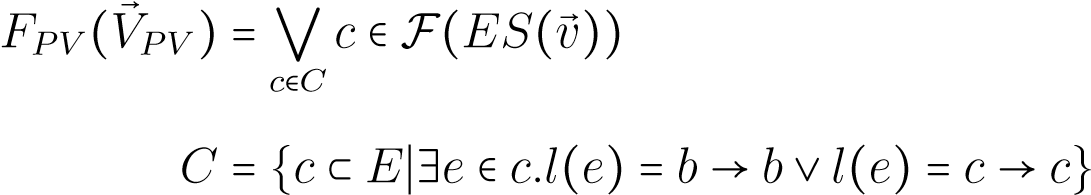
2

}

*p*2*,q*2 *q*2*,bb q*2*,cc*

شکل ۴. ۶

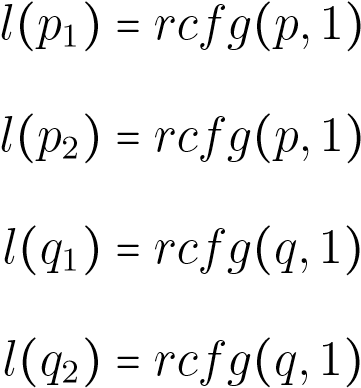
است پس از انجام به روز رسانی مسیر نارنجی امکان عملیاتی به شکل (*σb,σb*) وجود دارد که به معنی وجودحلقه در این شبکه است. اما اگر به روز رسانی مسیر سبز هم انجام شود تنها عملیات ممکن روی بسته ارسال آنبه سوییچ*d*  است. اکنون فرض کنید کهE =J*SDNPV*K ساختمان رویداد این شبکه وM مدل علیE  بر اساستعریف باشد. در این مدل تابع متغیر را به صورت زیر تعریف می کنیم:



در این تابع رفتار نا امن وجود پیکربندی ای شامل یکی از برچسب های*b* → *b*  یا*c* → *c*  در شبکه است. همانند( )مثال قبل با توجه به ترتیب اجرای به روز رسانی ها در شبکه دو رویداد برای هر یک از عملیات های (1*rcfg*(*p,* و

1*rcfg q,* در ساختمان رویداد وجود دارد. فرض کنید برای رویداد های مرتبط با این عملیات ها چهار رویداد

2*p*1*,p*2*,q*1*,q* وجود داشته باشد که برچسب آن ها به صورت زیر باشد:



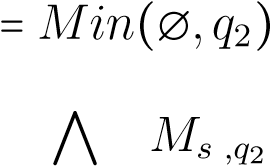
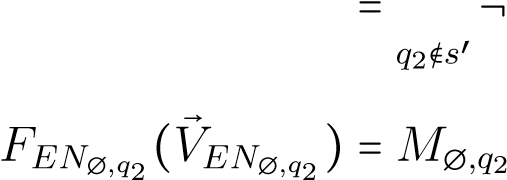
همچنین فرض کنید برچسب رویداد های*bb,cc*  به ترتیب*b* → *b,c* → *c*  باشد. شکل ۶ .۴ قسمتی از نمودارساختمان رویداد این شبکه را نشان می دهد که در آن تمام پیکر بندی هایی وجود داشته باشد که یکی از رویداد های

### ۵۷

*bb* یا*cc*  قابل دسترس باشد.

در این مثال میتوان*M*{*p*2}*,q*2 = False با در نظر گرفتنّ (*EN ,q*2 *M ,q*2 (∅*,*∅*,*Trueبه عنوان شاهد را یک علت واقعیوجود دور در این شبکه معرفی کرد. با توجه به تعریف مدل علی در بخش ۳ .۴ توابع متغیر های ∅ و ∅ به صورت زیر تعریف میشوند:

*FM*∅*,q*2(*V*⃗*M*∅*,q*2) = *Min*(∅*,q*2) ∧ *Con*(∅)

′

(2*EN*(∅*,q*با توجه به این توابع واضح است که اگر مقدار (2*M*({*p*2}*,q* را برابر صحیح قرار دهیم آنگاه مقدار (2*M*(∅*,q*غلط شده و در نتیجه مقدار هم غلط می شود. برای اینکه هر کدام از مجموعه های شاخه ی راستشکل ۶ .۴*p*2 *,q*2 Falseعضوی از مجموعه *v*⃗) *M*)ی پیکربندی *ES*های (⃗*ESW*⃗(*v*⃗) *ES*(*v*باشند باید داشته باشیم: 2*q* ⊢ ∅ اما با توجه به اینمقدار متغیر متناظر با این رابطه غلط شده است بنابراین این رابطه در وجود ندارد، پس هیچ کدام از اینمجموعه ها عضوی از پیکربندی های نیستند. بنابراین در این شرایط مقدار متغیر*PV*  غلط شده و شرط۲.آ در تعریف ۱ .۳ .۴ برقرار می شود. با توجه به گزارهی ۳ .۶ .۲ چون در شاهد خالی است می توان نتیجهگرفت که = ( { }) علت واقعی به وجود آمدن دور در این شبکه است.

## ۸ .۴ نبود سیاه چاله

در یک شبکه سیاه چاله ها۱۱ عناصری در شبکه هستند که وظیفه ارسال بسته ها را دارند (مثلا سوییچ یا روتر)ولی برخی از بسته ها را پس از دریافت به جایی ارسال نمی کنند و در واقع مانند سیاه چاله این بسته ها در آن هاگم می شوند [۳۰]. در یک شبکه که مکان های ورودی و خروجی مشخص دارد عدم وجود سیاه چاله در شبکه رامی توان معادل این ویژگی که همه ی بسته های ورودی به شبکه از آن خارج شوند دانست. به عنوان مثال شبکه ی

11Blackhole

### ۵۸

*s*

*b*

*a*

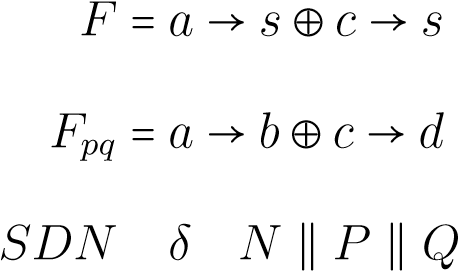
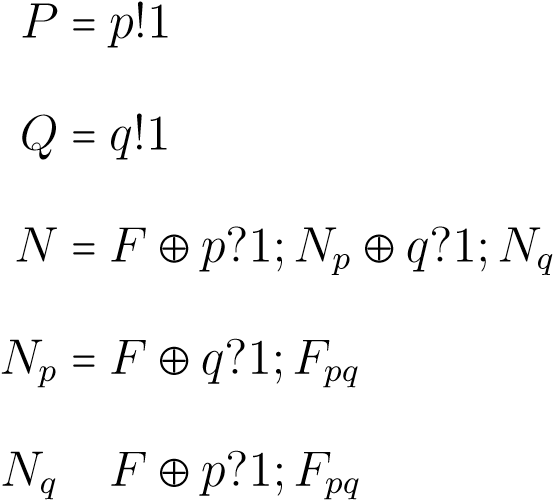
*d*

*c*

شکل ۴. ۷

موجود در شکل ۷ .۴ را در نظر بگیرید. فرض کنید که در این شبکه*a,c*  ورودی های شبکه و*s*  خروجی شبکهباشد. در این شبکه دو به روز رسانی برای جایگزینی مسیر*ds*  با*db*  و مسیر*bs*  با*bd*  انجام می شود. در این شبکهدر حالت ابتدایی و پس از انجام یکی از به روز رسانی ها ورودی ها به خروجی مسیر وجود دارد اما اگر هر دوی اینبه روز رسانی ها انجام شوند دیگر*s*  قابل دسترسی نیست و عملا بسته های ورودی به شبکه به خروجی نمی رسند.

این شبکه را می توانیم به فرم زیر در نت کت پویا توصیف کنیم:

= L*p*(!1*,p*?1*,q*?1*,q*)?1

= L= { }

در ادامه فرض کنید کهM  ّمدل علی این شبکه باشد. در این مدل تابع متغیر*PV*  را به صورت زیر تعریف میکنیم:

*FPV* (*V*⃗*PV* ) = ∃*c* ∈F(*ES*(*v*⃗))*,*∃*e* ∈ *c.l*(*e*) = *α* ⋅ *π* ∧ *π*(*sw*) ≠ *s*

*s*در تعریف این تابع رفتار نا امن وجود یک پیکربندی شامل رویدادی با برچسب از نوع*α* ⋅ *π*  یا به عبارت دیگررویدادی از نوع ارسال بسته است که سوییچ مقصد ارسال آن سوییچ نباشد. همانند مثال مربوط نبود دور

### ۵۹

∅

{

*p*

1

}

{

*p*

1

*,q*

1

}

{

*p*

1

*,q*

1

*,ab*

1

}

{

*p*

1

*,q*

1

*,cd*

1

}

{

*q*

2

}

{

*p*

2

*,q*

2

}

{

*p*

2

*,q*

2

*,ab*

2

}

{

*p*

2

*,q*

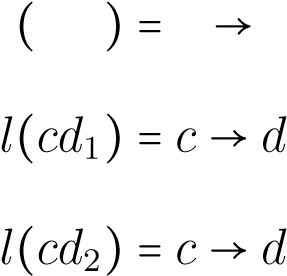
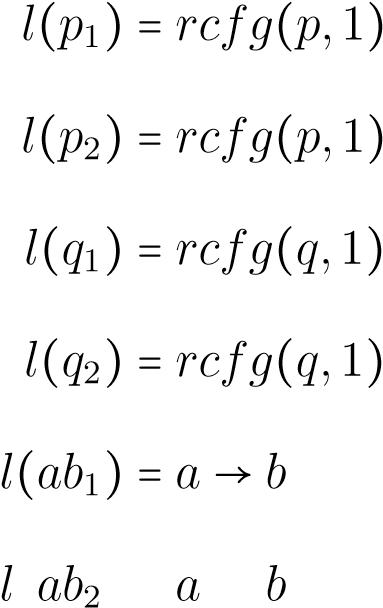
2

*,cd*

2

}

شکل ۴. ۸

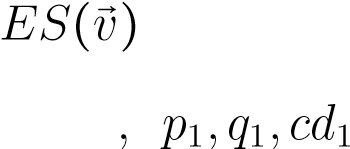
در شبکه، در این مثال هم با توجه به ترتیب اجرای به روز رسانی ها دو رویداد برای هر یک از عملیات های(1*rcfg*(*q,*1)*i*،=*rcfg*1*,*2(*p,* و*a* → *b,ac* → *d*  وجود دارد. بنابراین فرض کنید که رویدادهای*pi,qi,abi,cdi* به ازای در ساختمان رویداد این مدل وجود داشته باشند و برچسب گذاری آن ها به صورت زیر باشد:

پیکربندی *PV*هایی از این ساختمان رویداد که شامل رویدادی با برچسب*a* → *b*  یا*c* → *d*  باشند نقض شدن ویژگینبود سیاه چاله را نشان می دهند. بنابراین تابع متغیر را به فرم زیر توصیف می کنیم:

*FPV* (*V*⃗*PV* ) = ∃*c* ∈F(*ES*(*v*⃗))*,*∃*e* ∈ *c.l*(*e*) = *α* ⋅ *π* ∧ *π*(*sw*) ≠ *s*

### ۶۰

در این مدل هم همانند مثال نبود دور در شبکه عدم وجود تعارض بین رویداد های 1*p* و 1*q* را می توان به عنوان1*,q*1علت واقعی نقض شدن ویژگی در نظر گرفت*p*2*,q*2 .*p*برای این منظور از شاهد (*Cp*2*,q*2*,*True*,*True) استفاده می کنیم.واضح است که اگر مقدار هر دو متغیر*C*  و*C*  را برابر غلط قرار دهیم آنگاه هیچ یک از پیکر بندی های

{2*p*1*,q*1*,cd*1}*,*{*p*1*,qp*12*,q,ab*2 1}*,*{*pp*12*,q,q*1 2*,ab*2*p*}1*,,q*{1*p*=2*,q*(⃗2)*,cd*} {دیگرعضویازپیکربندی های} {1 1 1نیستند.}از طرفی در شرایطی که*C*  مقدار درست داشته باشد آنگاه پیکربندی های*p ,q ,ab*

عضو*ES v*  هستند و مقدار متغیر*C*  تاثیری روی این مساله ندارد بنابراین در نهایت می توان نتیجه گرفتکه*C* False  علت واقعی نقض ویژگی است.

۶۱

# فصل ۵جمع بندی و کا رهای آینده

## ۱ .۵ جمع بندی کا رهای انجام شده

دراینپژوهشروشیبرایاستفادهازتعریفعلتواقعیمطابق[۱۸]برایپیداکردنعلتخطادربرنامه هایتوصیف شده در زبان نت کت پویا ارائه شد.

برای امکان پذیر شدن استفاده از مدل علّی هالپرن و پرل از ساختمان رویداد به عنوان مدل معنایی برنامه هاینت کت پویا استفاده شد. سپس یک مدل علّی برای توصیف ساختمان رویداد در قالب معادلات ساختاری مطابق[۱۸] بیان شد که در آن رفتار نا امن شبکه در قالب معادله ی یکی از متغیر ها توصیف شده است. به صورت شهودیمدل علّی ساختمان رویداد این امکان را فراهم می کند که بتوان وجود رفتار نا امن را در ساختمان رویداد هایی کهناشی از اعمال تغییر در المان های ساختاری آن هستند را بررسی کرد، مثلا افزودن یک رابطه ی تعارض یا حذفیک رابطه ی فعال سازی. استفاده از این مدل معنایی همچنین این امکان را فراهم کرد که بتوان از تعریف علتواقعی ارائه شده توسط هالپرن و پرل به شکل مستقیم و بدون تغییر استفاده کرد. در نهایت با استفاده از روش ارائهشده چند نمونه از ویژگی های رایج شبکه مورد تحلیل قرار گرفتند تا علت واقعی نقض ویژگی در آن ها پیدا شود.همانطور که پیش تر در [۱۸] به آن اشاره شده، معیار مشخصی برای بررسی کیفیت یک فرمولاسیون علت واقعیوجود ندارد. به همین دلیل در این پژوهش هم صرفا میزان تطابق علت های پیدا شده با شهود موجود از مسالهبررسی شد.

### ۶۳

فصل ۵: جمع بندی و کار های آینده توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت

## ۲ .۵ نوآوری ها و دستاورد ها

### **۱ .۲ .۵ جستجو در ساختار**

همانطور که پیش تر بیان شد یکی از تفاوت های اصلی این پژوهش با پژوهش های پیشین مانند [۲۵، ۵، ۹]در زمینه ی توضیح خطا، امکان معرفی ساختار های سیستم، مثلا هم روند بودن دو عملیات، به عنوان علت خطابوده است. اما در پژوهش های پیشین رفتار های سیستم، مثلا یک دنباله از عملیات های سیستم، به عنوان علتخطا در نظر گرفته می شدند.

### **۲ .۲ .۵ استفاده از ساختمان رویداد به عنوان مدل معنایی**

در این پژوهش به جای استفاده از سیستم انتقال برای مدل معنایی برنامه های نت کت پویا از ساختمان رویداداستفاده شد. ساختمان رویداد همانند سیستم انتقال یک مدل محاسباتی برای پردازه های هم روند است. اما برخلاف سیستم انتقال که یک مدل برگ برگ شده است، ساختمان رویداد یک مدل غیر برگ برگ شده است. درمدل های برگ برگ شده، هم روندی پردازه ها با انتخاب غیرقطعی میان ترتیب های مختلف اجرای آن ها توصیفمی شود. به عنوان مثال دو پردازه ی*a*  و*b*  را در نظر بگیرید که به ترتیب یک عملیات*a*  و*b*  انجام داده و متوقفمی شوند. در مدل های برگبرگ شده رفتار*a* ∥ *b*  با رفتار*ab* + *ba*  معادل است، چون هم روندی پردازه ها صراحتادر مدل ذکر نمی شود. در طرف مقابل، در ساختمان رویداد این هم روندی به شکل صریح توصیف می شود. مثلادر مثال بالا نبود رابطه ی تعارض و فعال سازی بین رویداد های*a*  و*b*  به معنای هم روندی آن ها است که صراحتا درمدل قید شده است. یک از مزیت های استفاده از ساختمان رویداد در پژوهش جاری این است که امکان تعریفهم روندی دو عملیات به عنوان علت واقعی را فراهم می کند.

### **۳ .۲ .۵ استفاده مستقیم از تعریف علت**

بر خلاف کار های پیشین مانند [۸، ۲۵، ۶، ۹] که در آن ها از تعریف جدیدی از علت واقعی بر اساس تعریفHP استفاده شده است، در پژوهش جاری سعی شد تا مستقیما و بدون تغییر از تعریف علت واقعی ارائه شده در

[۱۸] استفاده شود. مزیت این رویکرد نسبت به استفاده از یک تعریف جدید این موضوع است که هنوز معیار

#### ۶۴

توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت فصل ۵: جمع بندی و کار های آیندهمشخصی برای مقایسه ی تعاریف علت واقعی وجود ندارد و به همین دلیل امکان ارزیابی تعریف جدید وجودندارد. در پژوهش جاری با استفاده از تعریف یک مدل علّی در قالب معادلات ساختاری این امکان فراهم شد تاتعریفHP مستقیما مورد استفاده قرار گیرد.

## ۳ .۵ محدودیت ها

### **۱ .۳ .۵ پیچیدگی زمانی**

یک ساختمان رویداد با*n*  رویداد را در نظر بگیرید. مدل علّی این ساختمان رویداد شامل (*O*(*n*2*n* متغیراست. برای پیدا کردن علت واقعی در این مدل و به طور خاص برای بررسی شرط ۲.ب لازم است تا تمامی زیرمجموعه های یک افراز از این متغیر ها بررسی شود که در بهترین حالت پیچیدگی زمانی (*O*(2*n*2*n* دارد. بنابراینپیاده سازی این روش بدون بهینه سازی یا استفاده از روش های ابتکاری عملا 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 نت کت پویا،لازم است تا رفتار نا امن در قالب یک تابع در مدل علّی و به عنوان یک شرط بر روی مجموعه ی پیکربندی هایساختمان رویداد منتج شده از آن توصیف شود. این مساله کار کردن با این روش را برای کاربر سخت می کند. راهحل مناسب ارائه یک منطق در سطح زبان است که به کاربر اجازه ی توصیف رفتار نا امن یا در روش بهتر اجازه یتوصیف ویژگی مورد نظر در قالب یک منطق را بدهد.

### **۳ .۳ .۵ استدلال در مورد یک علت**

روش ارائه شده در این پژوهش می تواند برای اثبات اینکه چه ساختاری از برنامه علت خطا است به کار رود.ولی این مساله به تنهایی برای تسهیل فرآیند اشکال زدایی سیستم کافی نیست. برای اینکه علت خطا بتواند به شکل

#### ۶۵

فصل ۵: جمع بندی و کارهای آینده توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیتکاربردی در فرآیند اشکال زدایی مورد استفاده قرار گیرد لازم است تا مشابه روش هایی مانند [۵] تمامی علت هایممکن برای خطا پیدا شوند و به کاربر نمایش داده شوند.

## ۴ .۵ کا رهای آینده

### **۱ .۴ .۵ ترکیب علت**

در [۹] نویسندگان ثابت کرده اند که امکان ترکیب علت ها در اجزای یک پردازه برای پیدا کردن علت در آنپردازه در شرایطی که پردازه ها ارتباط همگام دارند وجود ندارد. قدم بعدی این پژوهش اثبات امکان ترکیب علت هابرای پیدا کردن علت در یک پردازه ی بزرگ تر است. این مساله اولا تفسیر علت به دست آمده را ساده تر می کندثانیا می تواند باعث کاهش پیچیدگی زمانی پیدا کردن علت در یک پردازه ی مرکب شود.

### **۲ .۴ .۵ سنتز تعمیر**

با توجه به اینکه علت های پیدا شده در این پژوهش المان های ساختاری سیستم هستند، مثلا وجود هم روندیمیان دو عملیات، عملا این علت چگونگی رفع این مشکل در سیستم را نشان می دهد. مثلا اگر وجود هم روندیعلت به وجود آمد خطا در یک سیستم باشد برای رفع آن می توان یک ترتیب میان دو عملیات ایجاد کرد. اگرچگونگی پیاده سازی این ترتیب در سطح زبان نت کت پویا مشخص شود عملا می توان برای از علت خطا برایسنتز خودکار تعمیر برنامه استفاده کرد.

### **۳ .۴ .۵ مقایسه و رتبه بندی علت ها**

هالپرن و پرل در [۱۹] مفهوم مسئولیت۱ را در مدل علّی خود تعریف کرده اند. این مفهوم کمک می کند تابتوان میان علت ها تمایز قائل شد و یک معیار کمی به دست می دهد که بتوان علت ها را با یکدیگر مقایسه کرد. بهعنوان مثال یک انتخابات را در نظر بگیرید که در آن دو کاندیدا وجود دارد و کسی که اکثریت آرا از میان ۱۱ رای را

1Responsibility

#### ۶۶

توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت فصل ۵: جمع بندی و کار های آیندهکسب کند برنده انتخابات است. دو سناریو را در نظر بگیرید که در اولی نفر برنده با آرای ۶ به ۵ و در سناریوی دومبا آرای ۱۰ به ۱ برنده انتخابات می شود. واضح است که رای هر نفر به فرد برنده در سناریوی اول اهمیت بیشترینسبت به سناریوی دوم دارد چون برگرداندن رای هر نفر در سناریوی اول می تواند نتیجه ی انتخابات را تغییر دهد.در [۱۹] مفهوم مسئولیت به گونه ای تعریف شده است که به رای هر فرد به نفر برنده در سناریوی اول مسئولیتبیشتری، که یک مقدار عددی است، اختصاص می دهد. برای کمک گرفتن از میزان مسئولیت در این پژوهشمی توان پس از پیدا کردن چندین علت مختلف برای بروز خطا در یک شبکه، آن ها را بر اساس میزان مسئولیتشان مرتب کرد و سپس به کاربر نمایش داد تا کاربر راحت تر بتواند علت های مهم تر را شناسایی کند و از آن هابهره ببرد.

۶۷

# مراجع

1. Al-Shaer, Ehab and Al-Haj, Saeed. Flowchecker: Configuration analysis and verification of federated openflow infrastructures. In *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*, pages 37–44, 2010.
2. 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.
3. Aßmann, Uwe, Baier, Christel, Dubslaff, Clemens, Grzelak, Dominik, Hanisch, Simon, Hartono, Ardhi Putra Pratama, Köpsell, Stefan, Lin, Tianfang, and Strufe, Thorsten. Tactile computing: Essential building blocks for the tactile internet. In *Tactile Internet*, pages 293–317. Elsevier, 2021.
4. Baier, Christel, Dubslaff, Clemens, Funke, Florian, Jantsch, Simon, Majumdar, Rupak, Piribauer, Jakob, and Ziemek, Robin. From verification to causality-based explications. *arXiv:2105.09533 [cs]*, May 2021. arXiv: 2105.09533.
5. 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.
6. Caltais, Georgiana, Guetlein, Sophie Linnea, and Leue, Stefan. Causality for general ltldefinable properties. *Electronic Proceedings in Theoretical Computer Science*, 286:1–15, Jan 2019.
7. Caltais, Georgiana, Hojjat, Hossein, Mousavi, Mohammad, and Tunc, Hunkar Can. Dynetkat: An algebra of dynamic networks. *arXiv preprint arXiv:2102.10035*, 2021.
8. 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.

## ۶۹

1. 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.
2. 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.
3. Clarke, Edmund M. Model checking. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 54–56. Springer, 1997.
4. Clarke, Edmund M and Wing, Jeannette M. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996.
5. Datta, Anupam, Garg, Deepak, Kaynar, Dilsun, Sharma, Divya, and Sinha, Arunesh. Program actions as actual causes: A building block for accountability. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 261–275. IEEE, 2015.
6. Foerster, Klaus-Tycho, Schmid, Stefan, and Vissicchio, Stefano. Survey of consistent software-defined network updates. *IEEE Communications Surveys & Tutorials*, 21(2):1435– 1461, 2018.
7. Foster, Nate, Harrison, Rob, Freedman, Michael J, Monsanto, Christopher, Rexford, Jennifer, Story, Alec, and Walker, David. Frenetic: A network programming language. *ACM Sigplan Notices*, 46(9):279–291, 2011.
8. 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.

1. Halpern, Joseph Y. *Actual causality*. The MIT Press, Cambridge, Massachusetts, 2016.
2. Halpern, Joseph Y. and Pearl, Judea. Causes and explanations: A structural-model approach, part i: Causes. *arXiv:cs/0011012*, Nov 2005. arXiv: cs/0011012.
3. 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.
4. Hennessy, Matthew and Milner, Robin. On observing nondeterminism and concurrency. In *International Colloquium on Automata, Languages, and Programming*, pages 299–309.

Springer, 1980.

1. Khurshid, Ahmed, Zou, Xuan, Zhou, Wenxuan, Caesar, Matthew, and Godfrey, P Brighten. {VeriFlow}: Verifying {Network-Wide} invariants in real time. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 15–27, 2013.
2. 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.
3. Kozen, Dexter. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19(3):427–443, 1997.
4. Kreutz, Diego, Ramos, Fernando MV, Verissimo, Paulo Esteves, Rothenberg, Christian Esteve, Azodolmolky, Siamak, and Uhlig, Steve. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.
5. 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.
6. Lewis, David. *Counterfactuals*. Wiley, 1973.
7. 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. Monsanto, Christopher, Foster, Nate, Harrison, Rob, and Walker, David. A compiler and run-time system for network programming languages. *Acm sigplan notices*, 47(1):217–230, 2012.
9. Monsanto, Christopher, Reich, Joshua, Foster, Nate, Rexford, Jennifer, and Walker, David. Composing software defined networks. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 1–13, 2013.
10. 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.
11. Ruchansky, Natali and Proserpio, Davide. A (not) nice way to verify the openflow switch specification: Formal modelling of the openflow switch using alloy. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, pages 527–528, 2013.
12. Sassone, Vladimiro, Nielsen, Mogens, and Winskel, Glynn. Models for concurrency: Towards a classification. *Theoretical Computer Science*, 170(1-2):297–348, 1996.
13. Voellmy, Andreas and Hudak, Paul. Nettle: Taking the sting out of programming network routers. In *International Symposium on Practical Aspects of Declarative Languages*, pages 235–249. Springer, 2011.

توضیح خطا در شبکه های مبتنی بر نرم افزار با استفاده از استدلال مبتنی بر علیت فصل ۵: جمع بندی و کار های آینده

1. Voellmy, Andreas, Kim, Hyojoon, and Feamster, Nick. Procera: A language for high-level reactive network control. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 43–48, 2012.
2. Winskel, Glynn. *Event structures*, volume 255 of *Lecture Notes in Computer Science*, page 325–392. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.
3. Zeng, Hongyi, Zhang, Shidong, Ye, Fei, Jeyakumar, Vimalkumar, Ju, Mickey, Liu, Junda, McKeown, Nick, and Vahdat, Amin. Libra: Divide and conquer to verify forwarding tables in huge networks. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 87–99, 2014.

اول

## Abstract

The emerging use of Software-Defined Networks instead of traditional networking approaches have enabled us to use formal verification methods more easily on such systems which result in more reliable networks. But when an error is detected, the task of finding why the system doesn’t work as expected still remains and a person needs to manually investigate the problem and find out how to fix the system. In the case of failure, verification tools produce a certificate or counterexample but such evidence does not provide enough understanding of the problem.

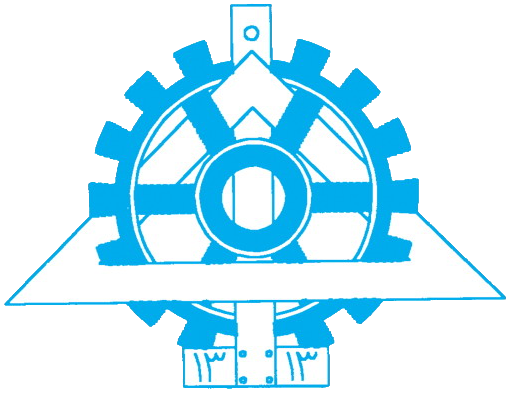
Explaining phenomena with causal reasoning has been studied for centuries in the philosophical literature which is distilled as the counterfactuality principle. Halpern and Pearl proposed a mathematical formulation of the actual cause based on Counterfactual reasoning. This formulation has been used in several research projects in order to use the actual cause of an error to provide an explanation of the system’s problem.

In this research, we use Halpern and Pearl’s notion of causality in the context of Software-Defined Networks to explain failures. More specifically we provide a causal model which can be used to reason about which constructs of the network, such as existence of concurrency or order between network updates, can be considered as an actual cause of the network’s unsafe behavior which results in a property violation.

To encode network programs we use DyNetKAT, which is a simple and high-level network programming language that can be compiled into the flow table of OpenFlow switches. DyNetKAT is built upon NetKAT and while preserving its minimality, it enables the encoding of dynamic updates in networks. We use event structures as a semantic model of DyNetKAT programs which allows us to explicitly encode concurrency and as a result, suchrelations may also be consideredas a cause while this is notpossible in an interleaving model. Weusedourmodeltoexplaintheviolationofsomewell-knownnetworkproperties by finding the actual cause of the failure. While there is no measurement or method to evaluate causal analysis, it seems that the causes found by our model matches the intuition of the root cause of the problem in the network.

**Keywords** Causal Reasoning, Event Structure, Formal Verification, Software-Defined

Networks

**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**

Supervisor:

**Dr. Hossein Hojjat**

Advisor:

**Dr. Mohammad Reza Mousavi**

September 2022